VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřící a řídicí techniky

Vestavěný řídicí systém pro detekci trajektorie pohybujícího se zařízení

Embedded System for Trajectory Detection of mobile Device

2010 **Ondřej Štrbík**

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě ……. Podpis studenta

Prohlašuji, že

jsem byl seznámen s tím, že na moji bakalářskou/diplomovou práci se plně vztahuje zákon č 121/2000 Sb. – autorský zákon, zejména §35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a §60 – školní dílo.

beru na vědomí, že Vysoká škola báňská – technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně ke své vnitřní potřebě bakalářskou/diplomovou práci užít (§35 ods. 3).

souhlasím s tím, že jeden výtisk bakalářské/diplomové práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a údaje o bakalářské/diplomové práci budou zveřejněny v informačním systému VŠB-TUO.

beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě ……. Podpis studenta

Poděkování

Rád bych tímto poděkoval vedoucímu bakalářské práce Ing. Zdeňku Macháčkovi, Ph.D. za poskytnutí teoretických podkladů a praktických rad. Dále pak mému kamarádovi, Bc. Radimovi Hercíkovi, který mi také poskytl cenné rady a nakonec své rodině za podporu při studiu.

Abstrakt:

Tato práce se zabývá návrhem a realizací vestavěného řídicího systému pro detekování trajektorie pohybujícího se mobilního zařízení. Hlavním úkolem je detekování předmětu známého tvaru a jeho vychýlení. V tomto případě se jedná o kolejnice autodráhy. Detekce vestavěného řídicího systému je realizována pomocí kamerového vývojového kitu. Navrhnuté a implementované algoritmy a metody analýzy použité v této práci, umožňují rozpoznat: zda se jedná o rovnou trajektorii dráhy, levotočivou zatáčku nebo pravotočivou zatáčku. Každá metoda byla testována v matematickém programu Matlab. Následně byl vybrán nejefektivnější a nejspolehlivější algoritmus detekce. Zjištěná trajektorie dráhy řídí rychlost vozu, tedy celého vestavěného systému, v dostatečném předstihu před následující změnou. Rozpoznání daného tvaru je základem pro úspěšné řízení zařízení.

Klíčová slova:

Kamera, Detekce trajektorie dráhy, Analýza obrazu, Metody rozpoznání obrazu, Autonomní řízení, Vestavěný řídicí systém

Abstract:

This work deals with designed and implementation of embedded control system for detection the trajectory of moving mobile device. The main task is to detect a known object shape and its deviation. In this case is a slot car track. Detection of embedded control system is realized by the development kit of camera.Designed and implemented algorithms and methods of analysis used in this study allows to identify: there are straight trajectory path, left-hand turn or right-hand turn. Each method was tested in the mathematical program of Matlab. Subsequently, it was selected the most efficient and reliable detection algorithm. Speed of the vehicle is controlled by an esteblished drack of trajectory, then the entire embedded kontrol systém, before the next change. Recognition of the shape is the basic for succesful managment of facilities.

Key word:

Camera, Tracks detection, Image analysis, Image recognition methods, Autonomous kontrol, Embedded control system

Seznam použitých symbolů a zkratek:

Bootloader	Zavaděč
CCD	Charge-Coupled Device, snímání obrazové informace
CC3	viz CMUcam3
CIF	Common Intermediate Format, rozlišení videa 352 × 288 v pixel
CMOS	Comlpementary Metal Oxid Semicondictor
CMUcam3	Kamera vývojového kitu
COM1	Rozhranní mezi počítačem a dalším zařízením
EABI	Embedded-application binary interface, konvence pro registry, datové typy
FAT16	File Allocation Table, souborový systém, verze FAT16
GCC	*GNU Compiler Collection, je sada překladačů včetně jazyka C*
GNU	Volně šiřitelný software, inspirovaný operačním systémem unixového stylu
GPIO	General Purpose Input/Output, je vstupně výstupní rozhraní zařízeních
HAL	Hardware Abstract Layer, softwarové rozhranní pro různý hardware
Hardware	Technické vybavení zařízení, například počítače
ISP	In Systém Programming, programování bez nutnosti vyjmutí
I2C	Inter-Integrated Circuit, počítačová sériová sběrnice
JPEG	Joint Photographic Experts Group, metoda ztrátové komprese obrázků
LED	Light – emitting diode
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
MMC	MultiMediaCard, standard paměťové karty s technologií paměti flash
PC	Personal computer, osobní počítač
QCIF	Quarter Common Intermediate Format, rozlišení videa 176 × 143 v pixel
RGB	Barevný model, červená – zelená – modrá
Software	Programové vybavení, vykonávající nějakou činnost
SD	Secure digital card – formát paměťové karty, velikost řádově GB
TTL	Transistor – transistor logic, tranzistorově-tranzistorová logika
UART	Synchroní a asynchroní seriové rozhranní
USB	Universal Serial Bus, univerzální seriová sběrnice

Obsah:

1. Úvod

Cílem bakalářské práce je realizovat vestavěný řídicí systém schopný detekovat určitý předmět známého tvaru a na základě výsledku reagovat. Zařízení se skládá ze samotné kamery, spolu s vývojovým kitem a elektronikou desky, která je umístěna na pohybujícím se zařízení. Tedy na vozidlu určeném pro autodráhu. Sledovaným předmětem se rozumí autodráha, zejména její vodící kolejnice.

Existuje mnoho druhů kamer určených pro robotiku. Využívají se jak černo bíle, tak i barevné verze těchto kamer. Vývojový kit použitý, jako součást vestavěného řídicího systému, využívá kameru vybavenou barevným rozlišení. Maximální možné rozlišení kamery je 352x288 pixel.

Pomocí kamery je pořízen snímek a následně se spustí algoritmus pro detekci, o jaký konkrétní předmět jde. Výsledkem je rovná trajektorie dráhy, nebo zatáčka a to pravotočivá či levotočivá. Tato informace je dále zpracována deskou vozu. Motor vozu je touto informací ovlivňován. Detekovaná rovná trajektorie dráhy znamená jízdu na maximum, naopak zatáčka znamená pomalou jízdu. Zjištěný obraz jednotlivých částí dráhy se ukládá na SD kartu. Slot pro SD kartu je standardní vybavení vývojového kitu.

V následující kapitole je popsáno seznámení se s kamerou a popis barevného modelu, se kterým kamera pracuje. Podrobně rozebrán princip a funkčnost kamery. Dále je zde zmíněno, s jakými programovými prostředky kamera a vývojový kit pracuje.

Seznámení se s modelem vozu a jeho elektronikou je popsáno ve třetí kapitole. Popsáno je zde také, s jakými programovými prostředky model vozu pracuje, respektive elektronika vozu. Naznačena implementace algoritmu do procesoru vozu.

Problematika metody analýzy obrazu jsou podrobně rozebrány v následující kapitole. Konkrétně jsou zde prováděny simulace různých možností a postupů zjištění trajektorie dráhy. Simulovány jsou také různé velikosti poloměrů zatáček.

Princip vestavěného řídicího systému a jeho realizace je rozebrána v další kapitole, což popisuje pátá kapitola. Program zjišťující a vyhodnocující tvar autodráhy je implementován v procesoru vývojového kitu kamery. Dále je zde popsán způsob propojení desky vozu a kamery. Také popis programu pro procesor desky, který zajišťuje reakci motoru na informaci z kamery. Konkrétně použitá řešení.

Kapitola věnující se testování celého realizovaného vestavěného systému popisuje testování a návaznost na jednotlivé části systému. Těmi jsou algoritmus detekce trajektorie dráhy použitý ve vývojovém kitu kamery s dalšími vyvinutými algoritmy detekce vytvořené v matematickém programu Matlab. Následně popsána uživatelská aplikace určená pro testování.

2. Zpracování a přenos dat z kamery

2.1 Kamerové zařízení

Kamera je součástí vestavěného řídicího systému pro analýzu obrazu. CMUcam3 je kamerové zařízení určené zejména pro robotiku a výzkum. Skládá se z malé videokamery a 32bit mikrokontroléru LPC2106 od NXP (Philips), se sériovým rozhraním. CMUcam má také velmi malé rozměry. Z těchto důvodů je poměrně využívaná pro výrobu malých mobilních robotů, pozorování, senzorové sítě, interaktivní hračky, rozpoznávání objektů a sledování.

C3088 je 1/4 " barevná kamera modulovaná s digitálním výstupem. Používá OmniVision CMOS obrazový snímač OV6620. Kombinací technologie CMOS, spolu se snadno použitelným digitálním rozhraním. Kamera umožňuje nízkonákladové řešení pro vyšší kvalitu obrazu a video aplikací.

Obr.1. Kamera [4]

Digitální video port dodává kontinuální 8/16 bit-široký obraz datového toku. Všechny funkce fotoaparátu, jako je expozice, gama, zisk, vyvážení bílé, barevné matice, jsou programovatelné přes rozhraní I2C. Tato kamera je určená pro programování zejména v operačních systémech Windows, tak i v operačním systémech Linux. Vlastní C kód, může být vyvinut pro použití GNU nástroje spolu se souborem open source knihoven. Jedná se o svobodný software unixového typu a také o volně přístupné knihovny pro dané zařízení. Spustitelné soubory mohou být nahrány pomocí sériového portu RS232.

Vstupní napájecí napětí je přivedeno na desku přes 5 voltový regulátor. Ideální napájecí napětí pro desku je mezi 6 a 15 DCV. Zdroj také musí dodávat minimálně 150 mA. Případně připojené servo pohony na portu, mohou být napájená přímo z desky nebo z externího zdroje přehozením propojky na desce. Pokud je vyžadován větší proud, než je maximálně povolený, dojde k resetu procesoru. Běh více servo pohonů nebude úspěšný. Porty, určené především pro servo pohony, lze použít také jako výstupy pro všeobecné digitální použití.

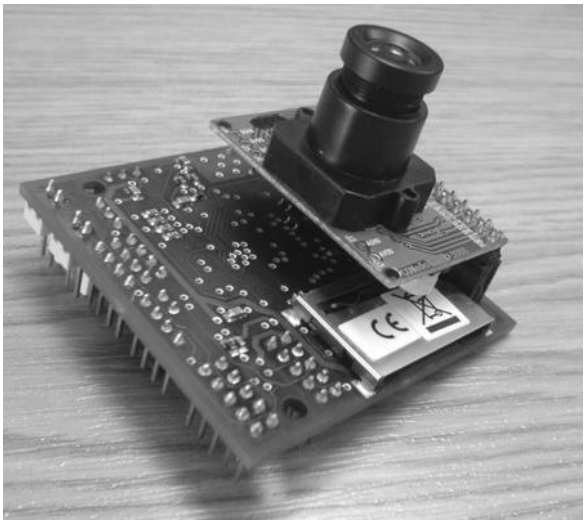
Funkce:

- Obsahuje progresivního snímání, což je metoda pro maximální ostrost obrazu
- Celkové rozlišení: 101376 pixel
- Formát CIF (352x288), nebo QCIF (176x143), což je daná velikostí obrazu
- Interní, nebo externí synchronizace systému
- Obsahuje funkce: jas, kontrast, gama, saturace, ostrost
- Konvoluční funkce pro zpracování obrazu
- Software komprese obrázku ve formátu JPEG
- Možnost emulace CMUcam2 kamery
- Elektronická expozice, zisku, vyvážení bílé barvy
- Ukládání dat ve formátu FAT16 na MMC, nebo SD kartu
- Možnost využití barevného modelu GBR nebo YCrCb
- Jednobarevný kompozitní video signál výstupu (50 Hz)
- Určené pro 8 / 16 bit

Upřesnění:

Vyobrazení	OV6620, CMOS obrazové čidlo
Počet snímků	26 snímků / s
CIF Rozlišení	352 x 288 pixel
Obrazový snímač	356 x 292 pixel
Velikost pixelů	9.0 x 8.2 µm
Viditelná oblast	3.1 x 2.5 mm
Poměr	S/N 48 dB
Minimální osvětlení	3lux @F1.2
Dynamické rozpětí	72 dB
Operační napětí	5 VDC
Spotřeba proudu:	16 mA
Čočka	f4.9 mm, F2.8 FOV=34.4x20.7°
	f3.7 mm, F2.0 FOV=43.7x25.8°

Vývojový kit je složen z modulu se seriovým rozhraním, nebo s rozhraním USB 2.0. Modul umožňuje nahrát video data získané z kamery do počítače. Elektronika desky je speciálně určena pro vyhodnocení digitálního obrazu pomocí OmniVision obrazového snímače. Vyhodnocovací senzor je do 5Mega Pixel. Rychlost přenosu je 48Mbps. Modul podporuje operační systém Windows a operační systém Linux. Další výhodou modulu s rozhraním USB je, že není vyžadováno externí napájení. Modul se seriovým rozhraním musí být externě napájen.



Obr. 2. Vývojový kit

Kamera je založena na technologii CMOS. Jde o technologii používanou ve většině integrovaných obvodů. Metal-oxid-semiconductor, odkazuje na fyzickou strukturu tranzistorů. Obsahuje kovovou řídící elektrodu. Kamera integruje obraz pole, zpracování signálu, načasování a ovládání obvodů, vše na jednom čipu. Použití jednoduchého obvodu nabízí jedinečné výhody, jako je nízká spotřeba, malé rozměry spolu s odpovídající kvalitou obrazu.

Kamera C-CamA a C-Cam2A mají stejné vlastnosti, liší se pouze jiným tvarem. Snímá černo-bíle, formát objektivu je 1/4", obrazové čidlo OV5116, objektiv f4.9mm, F2.8.

Kamera C-Cam8A je barevná, formát objektivu má 1/4". Používá obrazové čidlo OV7949, nebo OV6620. Objektiv f6.0mm, F1.8.

[2] [4]

Obr. 3. Různé druhy optik [4]

2.2 Fyzické propojení s PC

Sériový port je standardním vybavením kamery vývojového kitu, pro komunikaci s počítačem. Stejně jako TTL pro komunikaci s mikrokontrolérem. Vývojový kit využívá z celého sériového portu jen 3 piny z 10 pinů. Je v konfiguraci 2x5 pin. Odpovídá standardní 9-ti pinu plochého kabelu, sériové zásuvky a 10-ti pin sériové hlavičky. Sériová propojka musí být na svém místě, například během módu importování kódu do procesoru. TTL výstupní piny jsou mezi 0 a 3.3V, ale toleruje se i 5V.

Digitální vstupy / výstupy - je vstupně výstupní digitální rozhraní zařízení kamery. Umožňuje přístup ke druhé UART. UART je synchronní a asynchronní sériové rozhranní. Dále obsahuje různě řízené napájení na pinech a ISP pin.

Povolené napájení - Pokud je nízké napájecí napětí, hlavní regulátor zařízení způsobující povolení čerpání proudu od hodnoty 0.01uA, je vypnut. Veškeré zařízení je vypnuto a ztraceny veškeré informace. Pokud je odběr větší nastane opětovné spuštění. Deska se restartuje. Pin je ve výchozím nastavení.

AUX napájení – Tento pin může být nastaven pro napájení přímo z desky nebo z externě. Pin je standardně napájen 3.3 V. Při přesunutí rezistoru R11 a přidání propojky rezistoru na místo R6, pin bude připojen na 5 V napájecí napětí, před 5-ti voltový regulátor.

CAM RESET - Tento pin lze použít jako externí I / O, pokud stav kamery není nutné znát. Standardně tento pin resetuje modul kamery a neměl by být používán.

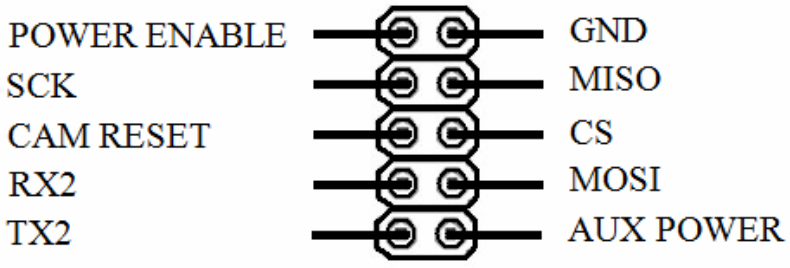
TX2 a RX2 – vysílací a přijímací pin na UART2 není na úrovni posunu, proto nemůže být přímo připojen k PC, neboť nemá TTL externí zařízení. Tento pin lze použít také jako GPIO.

CS – Slouží na znovuobnovení, jestli pin drží stav nízké úrovně, LPC2106 spustí zaváděcí režimu. Reboot může být externě vyvolaný pulzující energie umožněný pinem. Standardně je tento pin řízen MMC. Tento pin lze použít také jako GPIO nebo jako čip SPI, když MMC karta není vložena.

MOSI a MISO - Obvykle jsou tyto piny ovládány MMC. Lze je použít jako GPIO, nebo jako výstup SPI, pokud MMC karta není vložena.

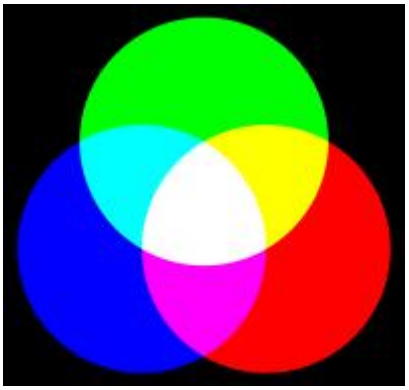
SCK - Obvykle tento pin je ovládán řadičem MMC. Tento pin lze použít také jako GPIO nebo jako SPI pin hodin, když MMC karta není vložena.

[2]



Obr. 4. GPIO port [2]

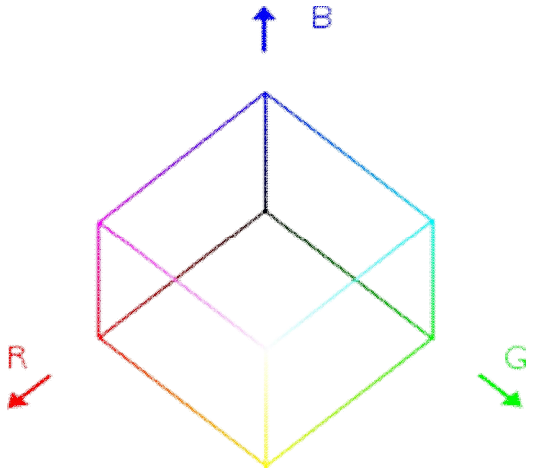
Princip zpracování barevného obrazu je založen na výběru tří základních barev. To má souvislost s fyziologií lidského oka. Drobné částice jsou podněty, které zvětšují rozdíl mezi reakcemi buněk lidské sítnice na světlo o různých vlnových délkách. Základní barvy tvoří barevný trojúhelník. Použití tří primárních barev není dostatečné, aby reprodukovalo všechny barvy. Pouze uvnitř trojúhelníka, mohou být reprodukovány aditivním mícháním nezáporného množství tohoto světla.



Obr. 5. Aditivní míchání barev [1]

Jedná se o aditivní míchání základních barev, kde se dvě barvy překrývají. Překrytím všech tří základních barev (červená, zelená, modrá) vznikne v odpovídající intenzitě bílá barva. Aditivním způsobem vytváření jednotlivých určitých barev se rozumí, barevný model RGB.

Název modelu je odvozen z počátečních písmen základních barev. Model sám o sobě nedefinuje, co je přesně myšleno červenou, zelenou a modrou barvou, proto výsledek smíchání složek není přesný, je relativní. Model je možné také zobrazit jako krychli. Každá z kolmých hran udává velikost intenzity barevných složek. Kterýkoli bod se souřadnicemi R, G, B. Představuje hodnotu výsledné barvy.



Obr. 6. Barevný model RGB [1]

Číselná zobrazení základních barev, které mají vlnové délky 630, 530 a 450nm. Velikost se udává dekadicky (procentuálně), nebo jako určitý počet bitů (barevná hloubka). Pro 8 bitů je rozsah 0 – 255, pro 16 bitů je rozsah 0 – 65535. Čím je velikost větší, tím s vyšší intenzitou se barva zobrazuje, je světlejší. Tento jev je znázorněn v barevném RGB modelu. Hodnoty v barevném modelu se zvětšují podél osy x (červená), y (zelená) a z (modrá). Pokud jsou veškeré barvy na minimální hodnotě, výsledkem je černá Naopak při maximální hodnotě všech barev, výsledkem je bílá.

Jednotlivé barvy jsou základní barvy v rozsahu 0 (minimum) a 1 (maximum). Mnoho rovnic používají následujících rozsahů. Plná intenzita červené barvy je 1, 0, 0. Hodnoty mohou být zapsány také procentuálně, 0% (minimum) 100% (maximum). Pro červenou to je 100% 0% 0%. Číselně v rozsahu od 0 – 255, to znamená pro červenou barvu hodnoty 255, 0, 0. Hexadecimální podobně pro tutéž barvu vychází FF, 00, 00.

R	G	B	BARVA
0	0	0	černá
255	0	0	červená
0	255	0	zelená
0	0	255	modrá
255	255	0	žlutá
255	0	255	purpurová
0	255	255	azurová
255	255	255	bílá

Tab. 1. 8 bit barevné spektrum v detailu [1]

Běžné použití tohoto RGB barevného modelu je rozdělení barevného elektronového paprsku, LCD displeje, plazmového displeje. Například monitor, nebo televize. Každý pixel na obrazovce může být reprezentován jako hodnoty pro červenou, zelenou a modrou. Hodnoty jsou převedeny do intenzity, nebo elektrického napětí přes gama korekci. Myšlená intenzita je reprodukována na displej. Běžné displeje využívají na 1 pixel 24 bitů, při 8 bitovém rozvržení. Každý pro červenou, zelenou a modrou barvu. Tento systém rozvržení dosahuje kombinace 16 777 216 (256^3 nebo 2^{24}).

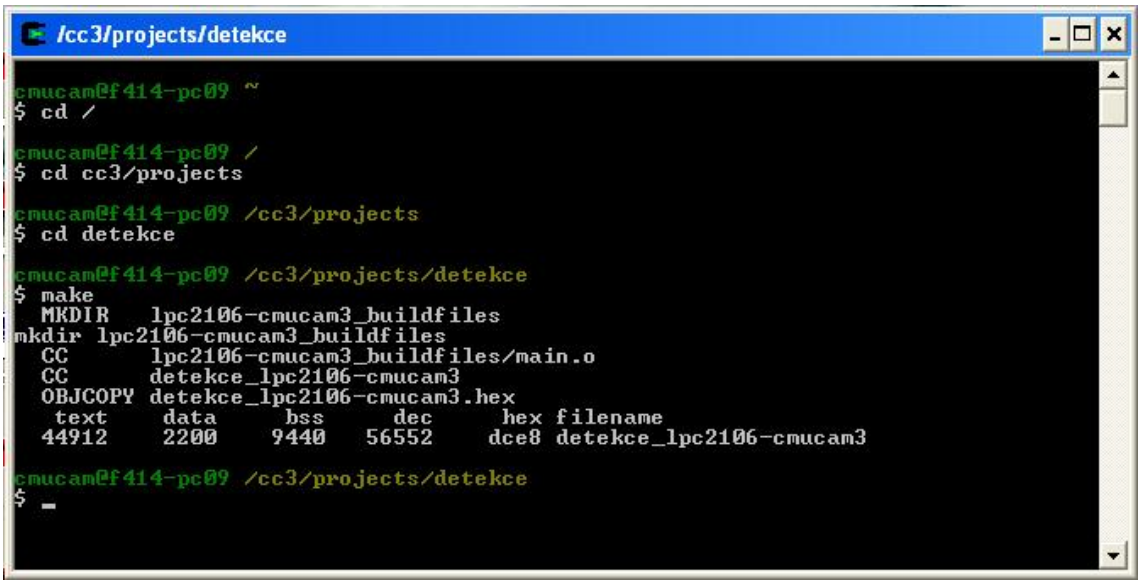
[1]

2.3 Programové prostředky kamery

Sestavení a implementace kódu určený pro kameru vestavěného řídicího systému je možné, jak v operačním systému Windows, tak i v operačním systému Linux. Pro správnou realizaci kódu je potřeba následující software pro sestavení vývojového prostředí:

- Cygwin
- ARM gcc
- LPC210x FLASH Utility

Cygwin je emulace operačního systému Linuxu pro prostředí operačního systému Windows. Obsahuje shell, což je překladatel příkazového řádku a další Linuxové nástroje. Například příkaz "make''. Tento příkaz vytvoří soubor s příponou hex, který se implementuje do vývojového kitu kamery. Používá se pro kompilaci kódu určený pro kameru. Pokud je použit systém Linux, většina těchto nástrojů je ve standardním nastavení a program Cygwin není potřeby. V příkazovém řádku emulátoru je orientace podobná, jako v prostředí operačního systému Linux, tudíž používané příkazy budou: CD (pro přesun do adresářů), nebo LS (zajištění výpisu na obrazovku aktuálního adresáře) atd.



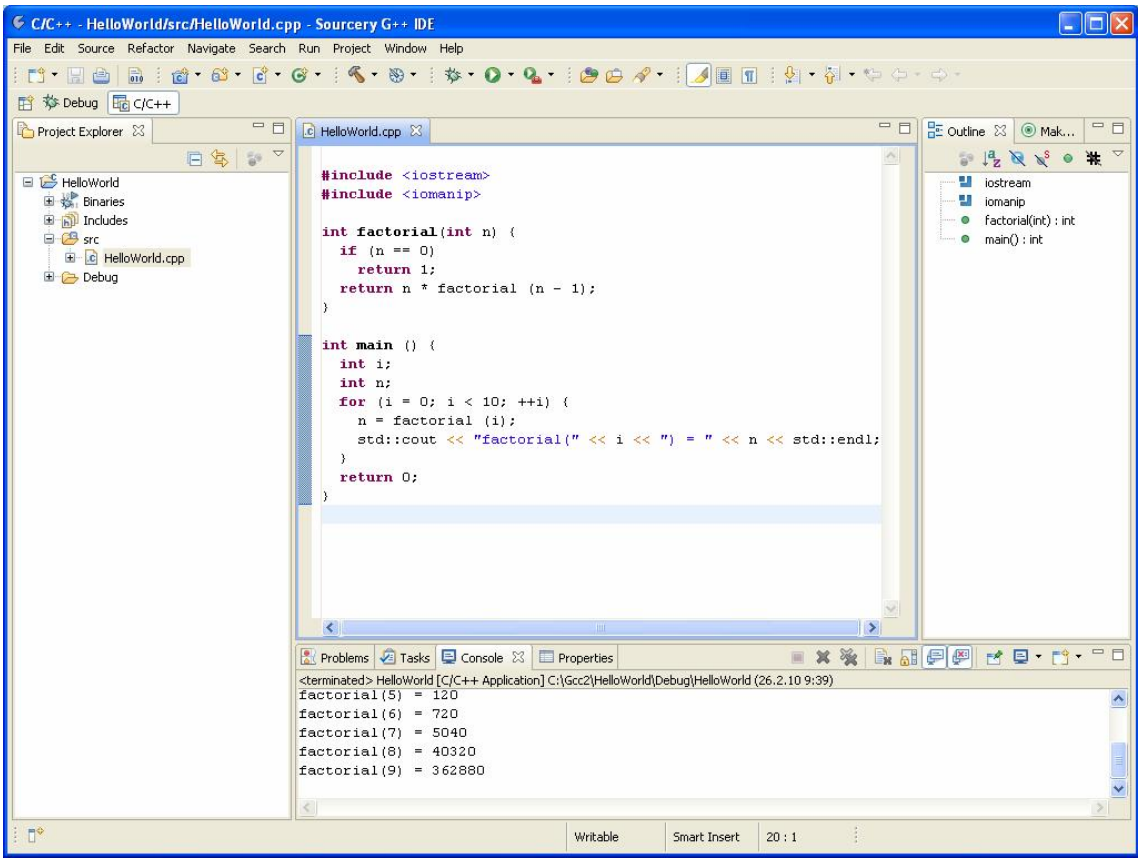
Obr. 7 Program Cygwin s příkazy

ARM gcc je kompilátor určený pro procesory ARM. Kompilátoru je ponechán výběr „hal". Hal obsahuje definici hardwaru. To umožňuje použití různých hardwarových platforem pro různé kompilátory. Podporován je hal pouze pro LPC2106, který používá verzi GCC. GCC představuje sadu překladačů včetně jazyka C. Optimalizován pro procesor ARM. Vytvoření programu, je potřeba použít vývojové prostředí s GCC nástroji.

Pro korektní fungování vývojového prostředí je potřeba instalovat následující. Hostitelská platforma: IA32 Windows a zároveň platforma EABI. Změna překladače se provádí změnou proměnné v defs.mk část COMPILER_PREFIX. Cesta umístění souboru je hal / lpc2106-cmucam3 / defs.mk. Lze změnit i jméno překladače, pokud je k dispozici vlastní.

Vlastní tvorba kódu pro kameru se provádí v programu Sourcery G++ IDE. Například ve verzi sourceryg++-4.4-70-arm-none-eabi. Tato verze programu je dostupná po registraci na stránce výrobce. Obsahuje grafické vývojové prostředí i kompilátor. Programuje se v jazyce C/C++. Pro použití je ovšem časově omezena a to po dobu 30 dní.

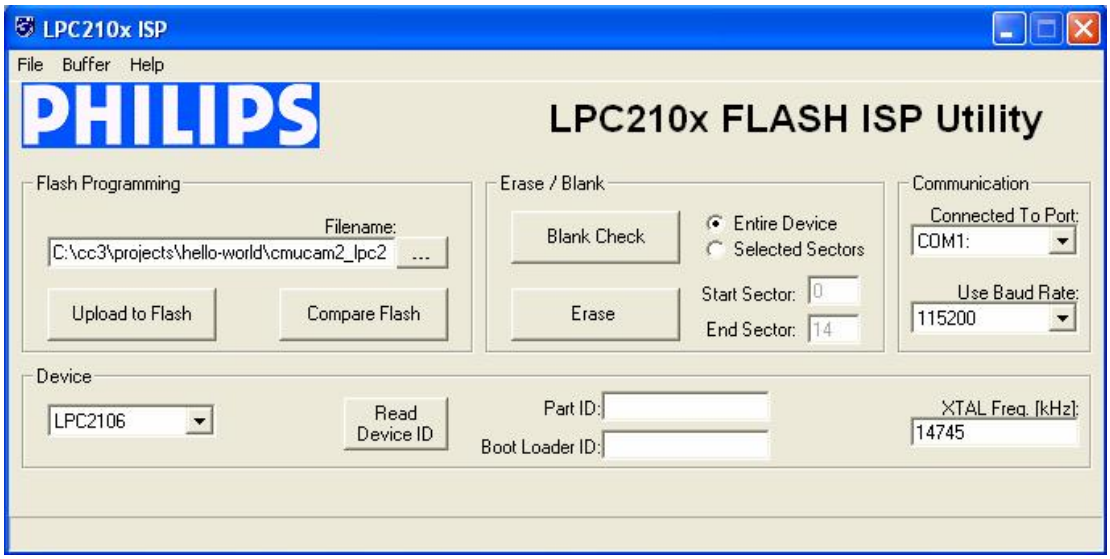
Alternativa ke tvorbě kódu kamery spočívá v použití dvou programů. Prvním je program obsahující kompilátor pro daný processor vývojového kitu, bez grafického prostředí. Například verze programu arm-2009q3-68-arm-none-eabi. Tato verze je volně dostupná. Druhý program slouží místo grafického prostředí. Lze použít pro tento účel například program Microsoft Visual Studio 2008.



Obr. 8 Vývojové prostředí program Sourcery G++ IDE

LPC210x Flash Utility, v tomto programu se provádí import vytvořeného kódu. V části „Flash Programming“ se vybere kód, který se nahraje, musí být ovšem s příponou hex. Jako první import se doporučuje zvolit program „hello_world_lpc2106-cmucam3.hex“. Jeho správné fungování je ověřeno. Dále se nastaví správná komunikace mezi PC a CMUcam3. Zahrnuje výběr com portu (COM1 nebo COM2), zařízení (LPC2106), frekvence (14745 kHz), přenosová rychlost (115200 puls/s). Kliknutím na tlačítko „Odeslat na Flash“, se odešle program do procesoru vývojového kitu. Pro ověření správného průběhu importu programu se objeví dialogové okno s hláškou „Nahrání souboru proběhlo úspěšně“.

Dalším předpokladem pro správné importování programu je, korektní resetování vývojového kitu. Provede se současným držením tlačítka ISP a přepnutím napájecího přepínače do polohy on. Tento úkon je potřeba provést pomalu, zvláště držet tlačítko po dobu 2 sekund. Mohli by dojít k neresetování procesoru na vývojovém kitu, spustil by se již dříve importovaný kód. Při správném resetování procesoru bude svítit zelená dioda napájení spolu se slabě červenou diodou. Kamera bude v režimu bootloaderu. Nastanou-li nějaké komplikace, je nutno tento postup zopakovat. Například je-li kamera už zapnuta, je potřeba ji vypnout.



Obr. 9. LPC210x ISP [3]

Struktura CC3 zdrojového kódu, jde o stromovou architekturu obsahující nejdůležitější části pro vytváření programů. Složka hal, v ní je uložen například název používaného překladače. Dále pak používané knihovny. Může se zde i ukládat vlastní kód.

Building systému. Jakmile bude kompilátor a zdrojové knihovny nainstalován, je potřeba vybudovat systém. Použije se příkaz „make“, který se zadá v příkazovém řádku programu Cygwin a to ve složce se zdrojovými kódy. Příkaz „make“ sestaví veškeré potřebné hal knihovny. Po spuštění programu se zadá: „cd c:“, tím se přejde na disk C. Dále pak do zdrojového adresáře pomocí: „cd my_directory“, postupuje se až do složky s názvem „CC3“. Příkaz „make“ použijeme v tomto adresáři. Vytvoří se potřebné knihovny.

[2] [3] [4]

3. Model vozu

3.1. Rozbor řízeného modelu vozu

Tento model s následnou nezbytnou úpravou (elektronikou), byl použit v prvním ročníku soutěže Freescale Race Challenge 2009. Jednalo se o autonomně řízené vozy, které ujedou 10 kol na obou kolejnicích autodráhy v jednom směru. Auta byla vybavena senzory různých tipů, minimálně však jedním a to akcelerometrem. Podle senzorů, nebo jediného senzoru se vozy řídily a mapovaly trať. Místo těchto nejrůznějších senzorů je v této práci použit kamerový vývojový kit. Má za úkol zabezpečit a zkvalitnit mapování dráhy. Model vozu se skládá ze dvou hlavních částí, desky a podvozku vozu.

Model je napájen z autodráhy stejnosměrným napětím 12V. Dotyk kolejnic s vozem pro napájení zajišťuje plovoucí vodítko FARO/28. Vůz je bez magnetu. Převodový poměr motoru je 3,75, jeho maximální otáčky jsou 12000 ot./min. Má 30 zubů na poháněné nápravě. Rozměry vozu jsou: délka 138mm, šířka 59mm výška 50mm.

Příklad vzorku dílu autodráhy je na Obr. 12. Povrch dráhy je přejiskřen, z důvodu lepší přilnavosti pneumatik vozu. Také tato úprava zmenšuje odraz nežádoucího světla dopadající na díly autodráhy. Jednotlivé díly zatáček mají poloměry: 222,5 mm, 265 mm, 402,5 mm a 447,5 mm.

Obr. 10 Model vozu [5]

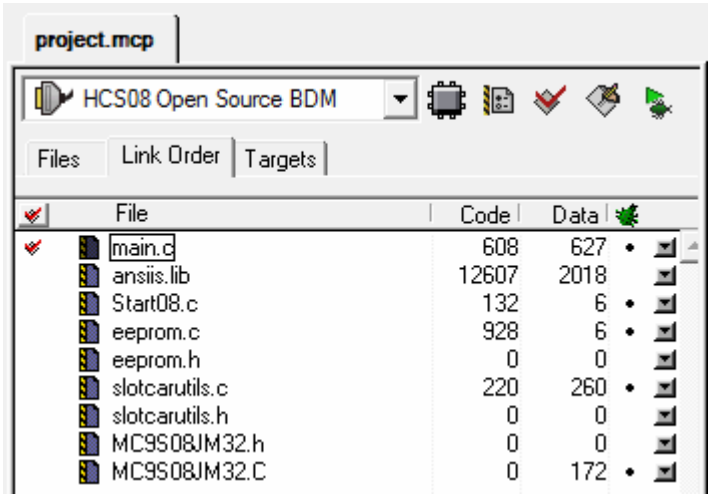
Obr. 11 Model dráhy [5]

Elektronika vozu je součástí vestavěného řídicího systému. Elektronika vozu je osazena 8 bitovým mikrokontrolérem S08JM32 od firmy Freescale. Také je zde akcelerometr, ten však zůstává nevyužit. Motor je ovládán přes H-můstek MC933887. Dále 4 led diody, sloužící pro informaci indikace zatáčky, nebo rovinky. Na desce je umístěna EEPROM paměť, určená pro ukládání dat. Pro připojení případných externích senzorů, je vyveden konektor umožňující toto rozšíření. Napájecí napětí 12V je převedeno na usměrňovač, stabilizátory napětí a vyhlazovací kondenzátory. Tím se na desku vozu dostane vždy správné napájení, pro všechny důležité část.

[5] [7]

3.2. Programové prostředky

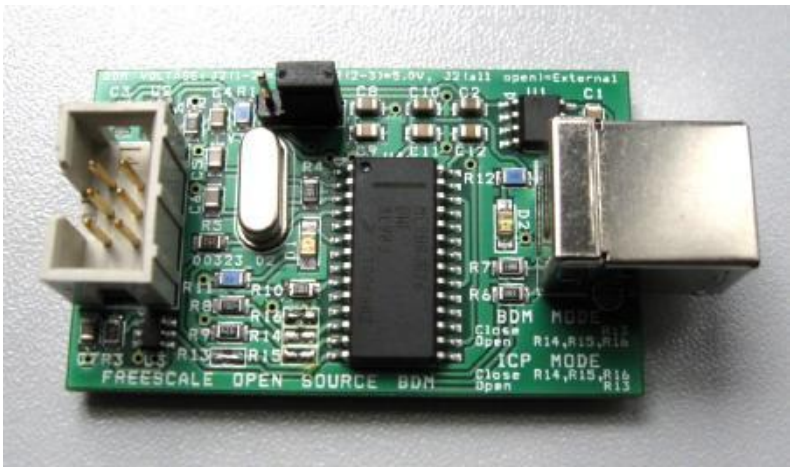
Procesor vozu se programuje v programovém prostředí CodeWarrior. CodeWarrior je integrované vývojové prostředí pro operační systémy Macintosh, Microsoft Windows, Linux, Solaris a vestavěné systémy. Původně byl vyvinut společností Metrowerks, nyní ji vyvíjí společnost Freescale Semiconductor. Existují i specializované verze, například pro Sony Playstation2. Je určen pro programování mikroprocesorů, mikrokontrolérů ColdFire. Hlavním programovacím nástrojem je využití jazyka C a C++. Také zahrnuje další programovací jazyky například: Pascal, Object Pascal, Objective C a Java kompilátory. Na Obr. 12 je zobrazena část vývojového prostředí se seznamem použitých částí kódu s ikonou spuštění implementace kódu do procesoru.



Obr. 12 Část vývojového prostředí

Vytvořený kód, pro daný procesor, je možno importovat do elektroniky vozu pomocí příslušného programátoru. Propojení programátoru s počítačem je realizováno pomocí USB rozhraní. Propojení programátoru s deskou vozu je realizováno pomocí BDM. BDM je rozhraní, které umožňuje ladění programů vestavěných systémů a nahrávání do paměti. Tento způsob ladění programu umožňuje i vyšetřování příčin případné chyby v běžícím programu. Díky programátoru je možno i číst z paměti data.

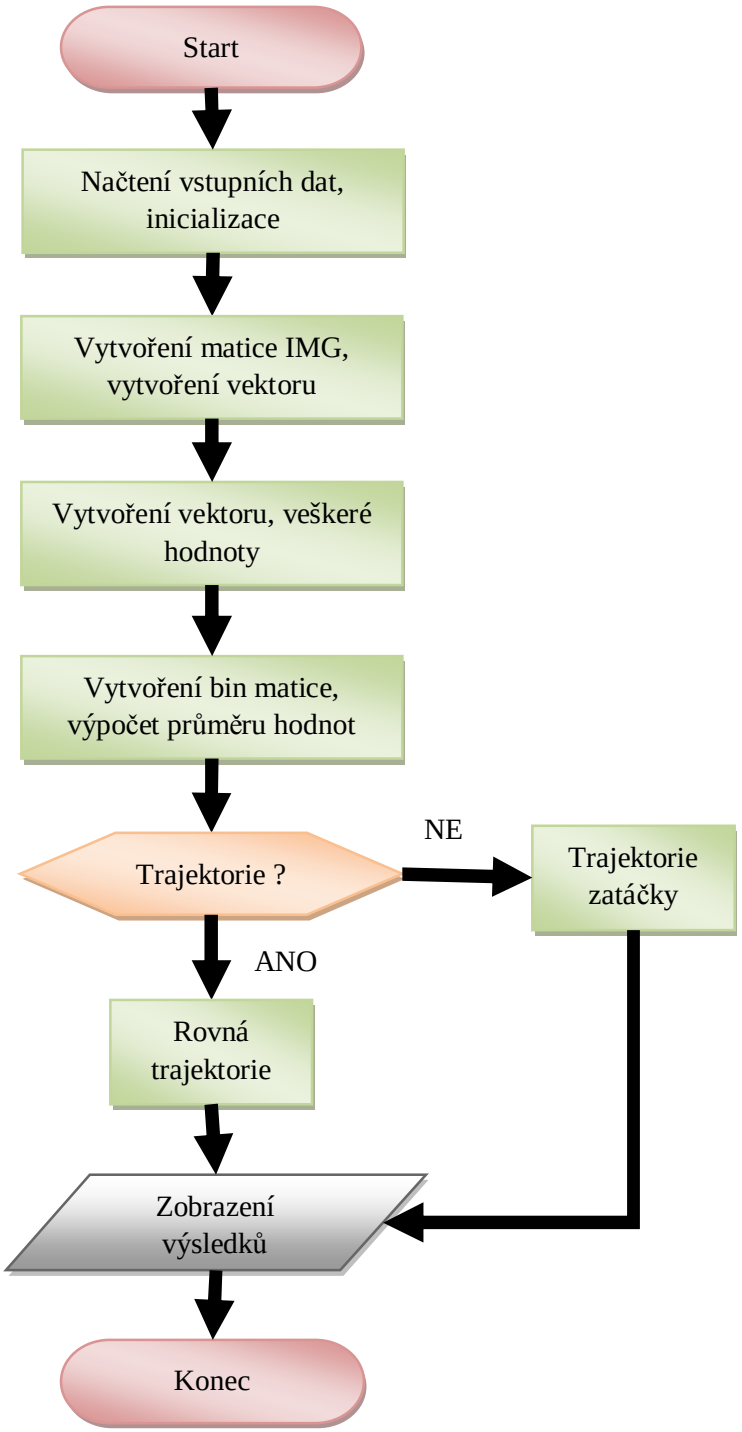
[9]



Obr. 13 Programátor [8]

4. Simulace algoritmu pro analýzu obrazu

Simulace v Matlabu byla vytvořena pro přiblížení problematiky detekce dráhy. Aplikace se potýká s několika zásadními postupnými problémy, které je nutno úspěšně vyřešit. Jinak výsledek detekce dráhy nemůže být dostatečně přesná. Prvotním problémem se rozumí, jak jsou data na snímku uložena, jak je vyčíst a dále zpracovat. Snímek obrazu je typu jpg, tudíž se využívá barevný model RGB. Model RGB obsahuje 3 základní barvy, červenou, zelenou, modrou. Každá základní barva reprezentuje matici dat. Program je zohledněn na jakoukoliv velikost obrázku. Byly vytvořeny dvě verze systému detekce pracující výhradně s maticemi.



Obr. 14 Vývojový diagram detekce pracující s maticemi

Základní nastavení a načtení snímku z kamery. Tato část programu je podobná ostatním simulacím algoritmů detekce dráhy.

```
clear all;close all;clc;warning off
obrazek = imread('2.jpg'); % nacteni dat z obrazku
obrazek = double(obrazek); %pretypovani z duvodu pouziti cisla nad 255
[y,x,z] = size(obrazek); % y,x,z jsou rozmery vstupniho snimku
```

První simulace je založen na sečtení všech tří základních matic a také se pracuje s výsledným vektorem. Dále pak je obraz zmenšen o 1/3 původní velikosti z důvodu zmenšení případné chyby. Vlastní detekce dráhy vychází z výpočtu středu kolejnic 5-ti horních a 5-ti spodních řádků vycházející z matice vektoruRR. Pro zjednodušení je maticeRR před touto operací převedena na binární matici. Výsledná podmínka je řešena pomocí funkce abs. Jednotlivé části kódu jsou okomentovány.

Inicializace proměnných. Část pro určení 1/3 osy x, tato hodnota se použije dále v programu.

```
suma_1=0; % hodnota pro stred kolejnic v horní casti obrazku
suma_2=0; % hodnota pro stred kolejnic v dolní casti obrazku
xx=0;
while xx < x/3  % 1/3 obrazku
    xx=xx+1;
end
```

Provádění dvojité smyčky určené pro získání hodnot jednotlivých matic R, G, B. Současně se tyto matice sečtou do matice img. Následně se vytvoří vektor s hodnotami vyskytujícími se v 1/3 na ose x.

```
for Y = 1:y % nastavovani radku
    for X = 1:x % indexi radku (sloupec)
        img(Y,X)=obrazek(Y,X,1)+obrazek(Y,X,2)+obrazek(Y,X,3);
    end
    vektor(Y,1)=img(Y,xx);
end% img je soucet vsech tri barev (RGB) v obraze
```

Další dvojitá smyčka je určena pro získání hodnot celého vektoru. Z matice img jsou odebírány hodnoty pouze od 1/3 až do 2/3 této matice. Vektor představuje křivkové součty hodnot v jediném řádku.

```
for X = 2:xx
    for Y = 1:y
        vektor(Y,X)=vektor(Y,X-1) + img(Y,X+xx);
    end
end
```

Tato dvojitá smyčka obsahuje podmínku určující převod na binární matici s názvem vektor_RR. Poté se vytvoří průměr z prvních 5-ti řádků a posledních 5-ti řádků z matice vektor_RR. Sečte se počet hodnot 1.

```
for X = 1:xx
    for Y = 1:y
        if vektor(Y,X) < vektor(Y,xx)/2
            vektor_RR(Y,X)=0;   % jde o bin matici
        else
            vektor_RR(Y,X)=1;
        end
    end
```

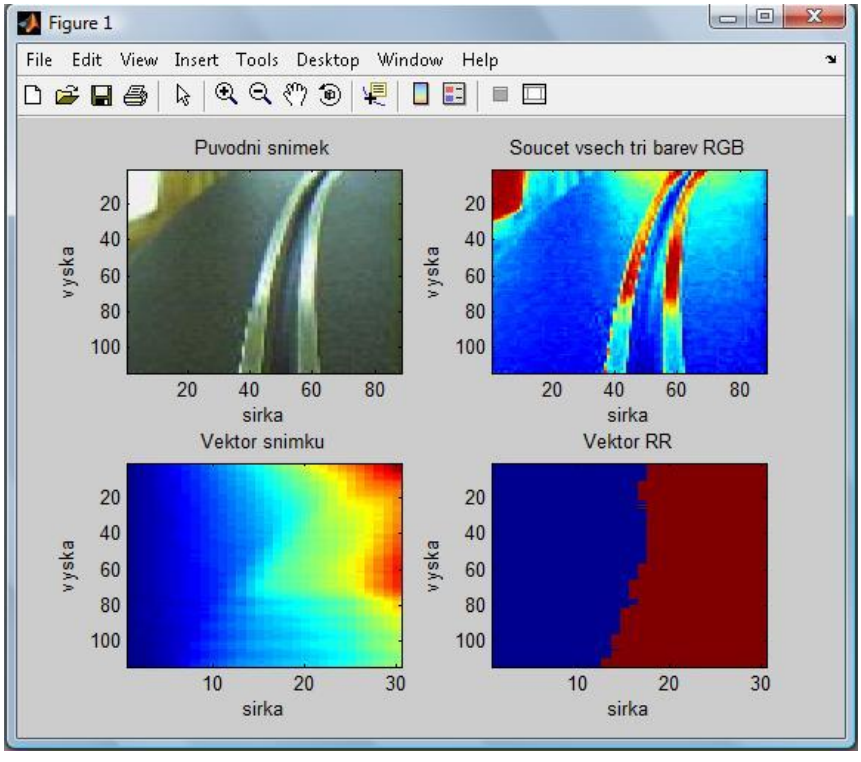
```
    % prvnich 5 radku
    if vektor_RR(1,X)==1   prvni radek horni casti
        suma_1=suma_1+1;   secitani poctu jednicek
    end   % ..atd.  % podobne pro dalších 5 radku, memi se osa Y

    % poslednich 5 radku
    if vektor_RR(y-1,X)==1   %prvni radek spodni casti
        suma_2=suma_2+1;   %secitani poctu jednicek
    end  % ..atd.  % podobne pro dalších 5 radku, memi se osa Y
end     % konec smycky for
```

Následující část obsahuje rozhodovací podmínku, zda jde o rovnou trajektorii dráhy či zatáčku. Pokud absolutní odečet sum je menší než hodnota 2, jedná se o rovnou trajektorii. Hodnota 2 určuje toleranci.

```
if abs(suma_1 - suma_2)<=2 %podmínka rozhodujici o trajektorii
    disp('Detekuje se rovna trajektorie');
else
    disp('Detekuje se trajektorie zatacky');
end
```

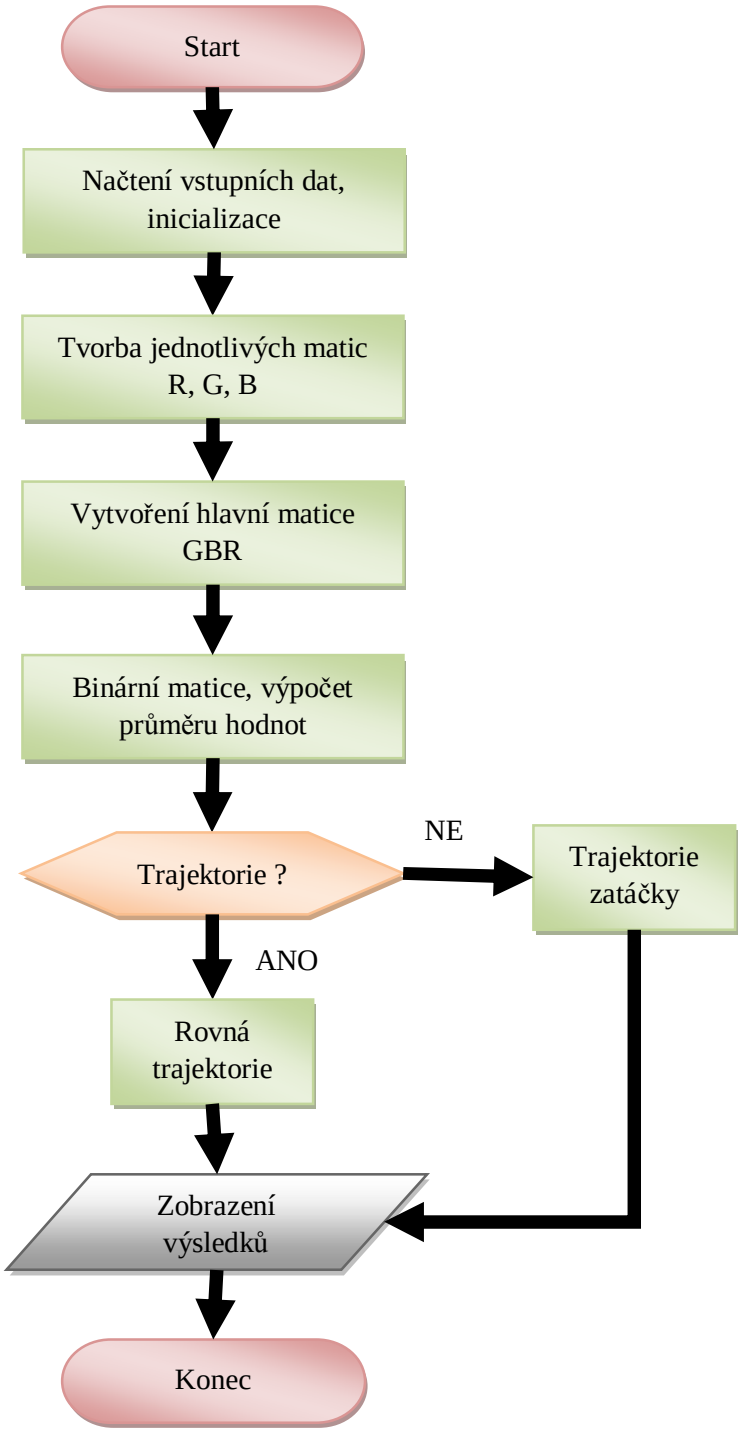
První simulace detekuje zatáčku. Jednotlivé zjištěné hodnoty jsou: suma_1 = 65 a suma_2 = 89. Při těchto hodnotách funkce ans zobrazí výsledek 24, což nesplňuje podmínku a jedná se o zatáčku. Tento výsledek platí pouze pro daný zpracovávaný obraz. Následuje vykreslení jednotlivých výsledků.

```
figure(1) % vykresleni jednotlivych vysledku
subplot(2,2,1) %Puvodni snimek
imagesc(obrazek); %imagesc(obrazek);
title('Puvodni snimek');
xlabel('sirka');
ylabel('vyska'); % atd…
```



Obr. 15 Zobrazení jednotlivých výsledku první simulace

Druhá simulace je založena na vytvoření tří základních matic. Tyto matice se od sebe odečtou a sečtou v daném pořadí: složka_barvy_G - složka_barvy_B + složka_barvy_R. Vznikne matice GBR. Na rozdíl od předchozí simulace, se zde s vektorem nepracuje. Detekce dráhy také vychází z výpočtu středu kolejnic. Řádky jsou vybrány opět 5 spodních řádků a 5 řádků v první třetině obrazu, pro zajištění eliminace možné chyby. Vytvoření binární matice je zajištěno konstantou v podmínce. Jednotlivé části kódu jsou okomentované. Následuje stručný výpis kódu.



Obr. 16 Vývojový diagram detekce pracující s maticí

Dvojitá smyčka pro vytvoření tří matic. Každá matice obsahuje jednu složku základní barvy. Před smyčkou je ještě inicializace proměnných.

```
suma_bin1=0;
suma_bin2=0;

for X = 1:x %pouze indexy        %selekce slozek R G B, 3 matice
    for Y = 1:y %pouze indexy
        slozka_barvy_R(Y,X) = obrazek(Y,X,1);
        slozka_barvy_G(Y,X) = obrazek(Y,X,2);
        slozka_barvy_B(Y,X) = obrazek(Y,X,3);
    end
end
```

Následuje vytvoření matice GBR. Vznikne odečtením složky barvy B od složky G s přičtením složky barvy R.

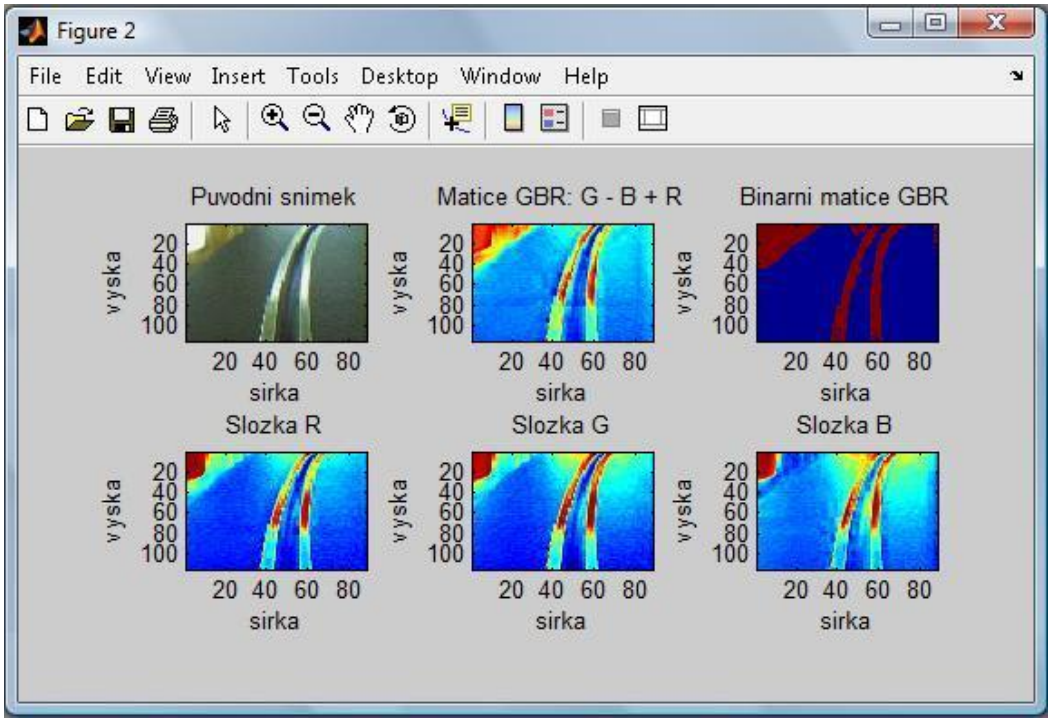
```
GBR=slozka_barvy_G - slozka_barvy_B + slozka_barvy_R; % secteni matic
```

Následuje dvojitá smyčka obsahující podmínku pro převod na binární matici. Při současném provádění obou smyček se také vypočítává střed kolejnic horní a spodní části snímku. Horní část je až od 1/3 snímku z důvodu zmenšení případné chyby.

```
% stred je pocitan jako prumer z 5-ti radku v 1/3 obrazku a dolnich 5-
ti radku, kvuli minimalizaci mozne chyby
for X = 1:x    % osa x
    for Y = 1:y
        if GBR(Y,X)> 127    % konstantni hodnota pro podminku 0/1
            bin(Y,X)=1;   % jde o bin matici
        else
            bin(Y,X)=0;
        end
    end

    % 5 radku pro vypocet stredu, nachazejici se v 1/3 obrazku
    if bin(95,X)==1
        suma_bin1=suma_bin1+1;
    end  % ..atd.  % podobne pro dalších 5 radku, meni se osa Y o 1

    % poslednich 5 radku
    if bin(y-1,X)==1
        suma_bin2=suma_bin2+1;
    end  % ..atd.  % podobne pro dalších 5 radku, meni se osa Y o 1
end     % konec smycky for
```

Nasleduje část pro výslednou podmínku určení trajektorie dráhy. Jednotlivé zjištěné hodnoty jsou: suma_bin1 = 81 a suma_bin2 = 53. Při těchto hodnotách funkce ans zobrazí výsledek 28, což splňuje podmínku a jedná se o trajektorii zatáčky. Tento výsledek platí pouze pro daný zpracovávaný obraz. Následuje zobrazení jednotlivých matic.

```
% vypsani vysledku na obrazovku, vysledek = akce
abs(suma_bin1 - suma_bin2)
if abs(suma_bin1 - suma_bin2)<=2
    disp('Detekuje se rovna trajektorie');
else
    disp('Detekuje se trajektorie zatacky');
end
```



Obr. 17 Zobrazení jednotlivých výsledku druhé simulace

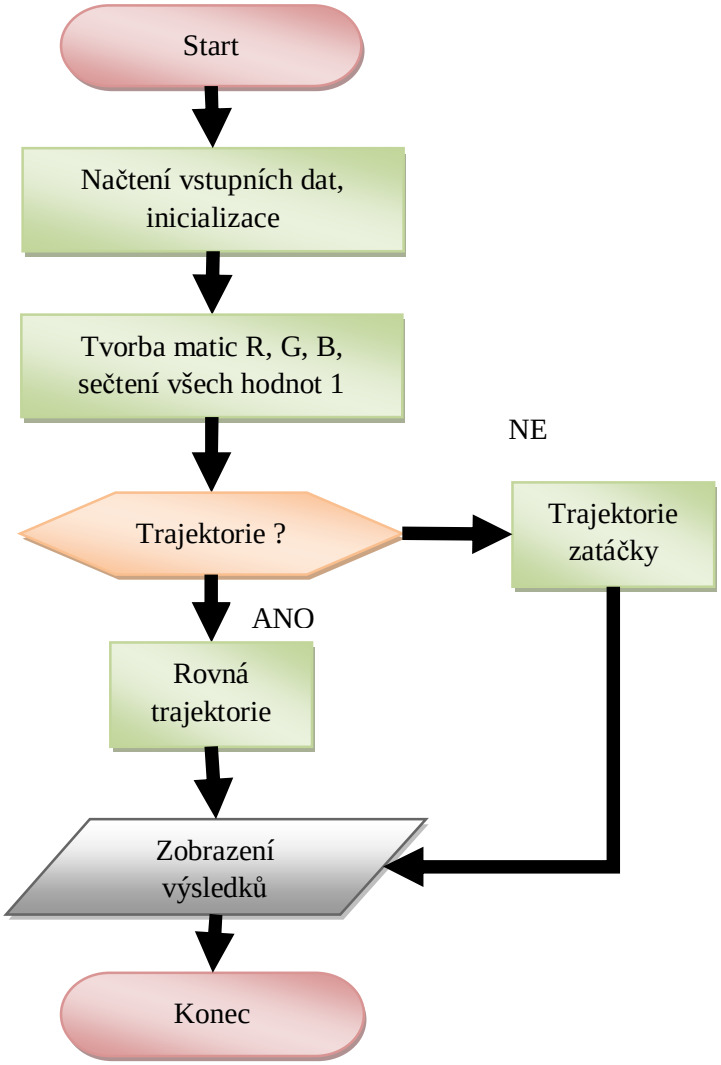
Další možností je detekování z určité části obrazu. Plné rozlišení obrazu je 176x143, to znamená celkově 25168pixel. Kolejnice dráhy jsou umístěné s nepatrnou odchylkou uprostřed obrazu. Z hlediska detekce okraje obrazu jsou nezajímavé, obsahují jen černý povrch autodráhy. Došlo tedy k oříznutí o ¼ z každé strany v ose x. Rovněž došlo k oříznutí v ose y, z důvodu dopadu světla na vzdálenější část od kamery. To způsobuje odraz od části autodráhy, může nepříznivě ovlivňovat správnou detekci. Stejný případ je u zatáčky, kdy kamera snímá i okraj, kde není dráha s kolejnicemi. Výsledná zkoumaná plocha je: 96x88, což je 8448pixel. Následuje stručný výpis kódu.

Základní nastavení pro program Matlab, zavření všech otevřených oken, vymazání všech proměnných, vymazání vytisknutých příkazů v Command Window a vypnutí varování.

```
clear all;  % vymazani vsech promennych
close all;  % zavreni vsech otevrenych
clc;    % vymazani vytisknutych prikazu v Command Window
warning off % vypnuti varovani
```

Načtení obrazu získané z kamery o rozměru 176x143, přiřazení obrazu do proměnné. Dále pak vynulování proměnné bod.

```
img = imread('02.JPG');  % nacteni dat ze snimku kamery
bod = 0;    % inicializace
```



Obr. 18 Simulace detekce pracující s maticemi

Provádění dvojité smyčky pro osu x a osu y. Tato dvojitá smyčka prochází zvolenou plochu obrazu z výše popsaných důvodů. Následují podmínky pro každou složku barvy zvlášť. Každá podmínka je zaměřená na velikost hledaného čísla, což je číslo 127. Tím se určí pomyslná 1, následuje řádek pro sčítání výskytu jedniček.

```
% maximalni velikost zkoumane plochy je 8448pixel
% zmenseni z duvodu nepotrebnosti okraju, jsou vzdy stejne a to cerne
for x = 44:132 % osa x zmensena o 2/4
    for y = 47:143   % osa y zmensena o 1/3

        if img(y,x,1) > 127   %matice slozka barvy R
            bod = bod +1;   % pocet jednicek v matici R
        end
        if img(y,x,2) > 127   %matice slozka barvy G
            bod = bod +1;   % pocet jednicek v matici G
        end
        if img(y,x,3) > 127   %matice slozka barvy B
            bod = bod +1;   % pocet jednicek v matici B
        end
    end
end
```

Tato část kódu obsahuje rozhodovací podmínku, zda jde o rovnou trajektorii dráhy, či o zatáčku. Proměnná bod v předchozí dvojité smyčce spočítala celkový výskyt jedniček a na základě této hodnoty se určí výsledek, v tomto případě jde o rovinu. Vykreslení obrazu, určeného pro detekci.

```
bod % tisk poctu jednicek
%rozhodovaci podminka
if bod > 2000 % hranice poctu jednicek
    disp('Detekuje se rovna trajektorie')
else
    disp('Detekuje se trajektorie zatacky')
end


figure(1)   % vykresleni nacteneho snimku
imagesc(img)
```

Výtisk výsledku na obrazovku v programu Matlab.

bod = 2461

rovinka

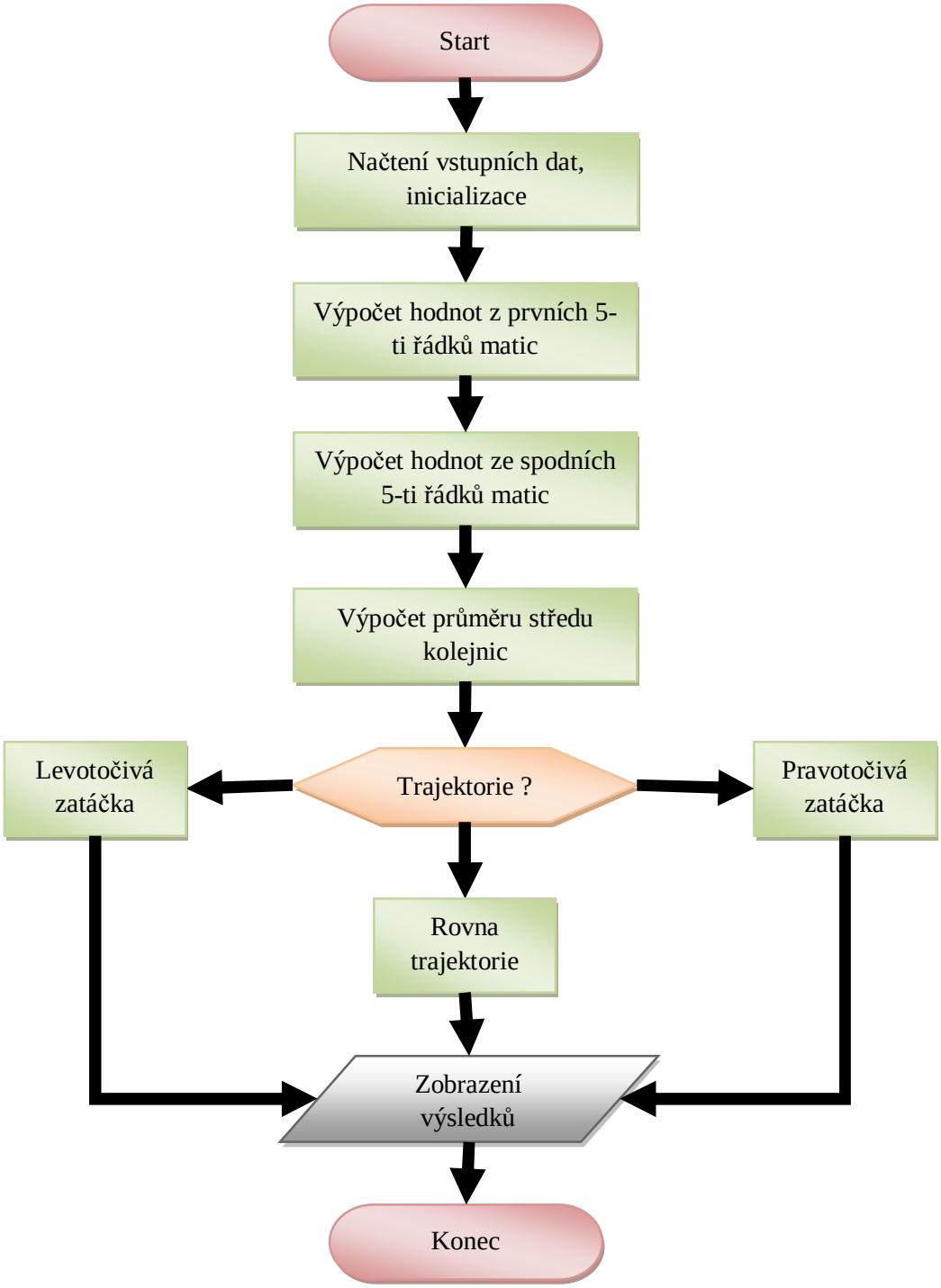
Podobnou možností je detekce založená na určení středu kolejnic. Program hledá střed kolejnic umístěný ve spodní části a ve vrchní části snímku. Hledá se souřadnice na ose x. Se získanými těmito hodnotami se dále pracuje v podmínce určení trajektorie dráhy. Rozlišení vzorku snímku je 352x287 pixel. Došlo k oříznutí osy y o 1/3, z důvodu dopadu světla na vzdálenější část od kamery. To způsobuje odraz od části autodráhy, může nepříznivě ovlivňovat správnou detekci. Následuje stručný výpis kódu.

Základní nastavení pro program Matlab, zavření všech otevřených oken, vymazání všech proměnných, vymazání vytisknutých příkazů v Command Window a vypnutí varování.

```
clear all;  % vymazani vsech promennych
close all;  % zavreni vsech otevrenych
clc;    % vymazani vytisknutych prikazu v Command Window
warning off % vypnuti varovani
```

Načtení obrazu získané z kamery o rozměru 352x287, přiřazení obrazu do proměnné. Vykreslení obrazu, určeného pro detekci. Dále pak vynulování všech používaných proměnných.

```
obrazek = imread('5.jpg');  % nacteni dat ze snimku kamery
vykr_obrazek = obrazek; % promenna pouze pro zobrazeni

bod = 0;    % inicialize vsech pouzitych promennych
bodx = 0;
vyslHorni = 0;
bod1 = 0;
bod1x = 0;
vyslSpodni = 0;
```



Obr. 19 Vývojový diagram zjištění středu kolejnic

Následuje část kódu pro určení středu kolejnice ve vrchní části snímku. Obsahuje dvě smyčky for pro osu y a osu x. Z celé plochy snímku, se pracuje s prvními pěti řádky. Spočítá se celkový počet jedniček a zároveň se provede sečtení všech indexů x, kde se jedničky nachází. Definice jedniček se nachází v podmínce pro každou matici zvlášť.

```
% vrchni cast snimku

for x = 25:146 % osa x plna velikost
    for y = 47:51   % osa y zmensena o 1/3, vyber 5-ti hornich radku
        if obrazek(y,x,1) > 127   %matice slozky barvy R
            bod = bod +1;   % pocet jednicek v matici R
            bodx = bodx +x;   % secteni hodnot na ose x
        end
        if obrazek(y,x,2) > 127   %matice slozky barvy G
            bod = bod +1;   % pocet jednicek v matici G
            bodx = bodx +x;   % secteni hodnot na ose x
        end
        if obrazek(y,x,3) > 127   %matice slozky barvy B
            bod = bod +1;   % pocet jednicek v matici B
            bodx = bodx +x;   % secteni hodnot na ose x
        end
    end
end
```

Druhá dvojitá smyčka for, je určena pro zjištění středu kolejnice ve spodní části snímku. Opět se pracuje s pěti řádky. Osa y začíná na hodnotě 283. Zbytek této části kódu je podobný jako předcházející část. Výpočet průměru středu kolejnice z vrchní a spodní části snímku.

```
vyslHorni = bodx/bod  % vypocet prumeru stredu kolejnice, horni cast
vyslSpodni = bod1x/bod1  % vypocet ze spodni casti snimku
```

Následující část obsahuje rozhodovací podmínku, zda jde o rovnou trajektorii či zatáčku. Proměnné vyslHorni a vyslSpodni obsahují informace se kterými se dále pracuje. Tyto dvě hodnoty se porovnávají a na jejich základě se určí výsledek.

```
if (vyslSpodni == vyslHorni) % vysledna podminka
    disp('Detekovana rovna trejektorie drahy');
elseif (vyslHorni > vyslSpodni)
    disp('Detekovana trajektorie zatacky VPRAVO');
elseif (vyslSpodni > vyslHorni)
    disp('Detekovana trajektorie zatacky VLEVO');
end
```

Výtisk výsledku na obrazovku v programu Matlab a vykresleni zkoumaného vzorku snímku. Vyšlo: vyslHorni = 283,4022 a vyslSpodni = 213,1543. Podle podmínky byla detekována trajektorie zatáčky vpravo.

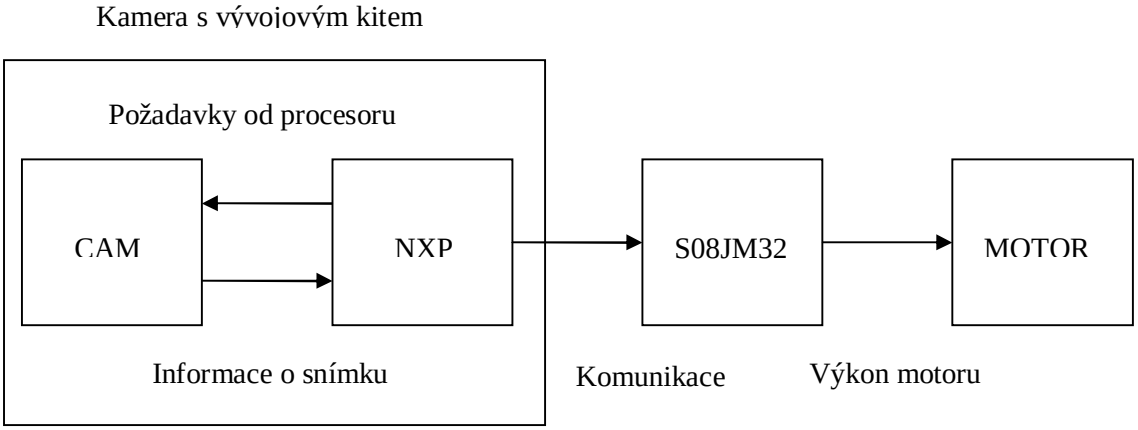
```
vyslHorni = 283.4022
vyslSpodni = 213.1543
Detekovana trajektorie zatacky VPRAVO
```

```
figure(1) % vykresleni nacteneho snimku
imagesc(vykr_obrazek);
title('Puvodni snimek');
xlabel('Osa x');
ylabel('Osa y');
```

5. Propojení kamery a SW

5.1. Princip vestavěného řídicího systému

Vestavěný řídicí systém se skládá ze dvou hlavních částí, těmi jsou kamera s vývojovým kitem a elektronikou vozu. Tyto dvě hlavní části systému jsou určeny pro autonomní řízení vozu na dráze. Další důležitou součástí je propojení mezi těmito částmi systému. Systém je složen ze dvou procesorů různých výrobců. Procesor na vývojovém kitu je od firmy Filips a elektronika vozu je osazena procesorem od firmy Freescale. Každý z těchto procesorů se programuje v jiném vývojovém prostředí.



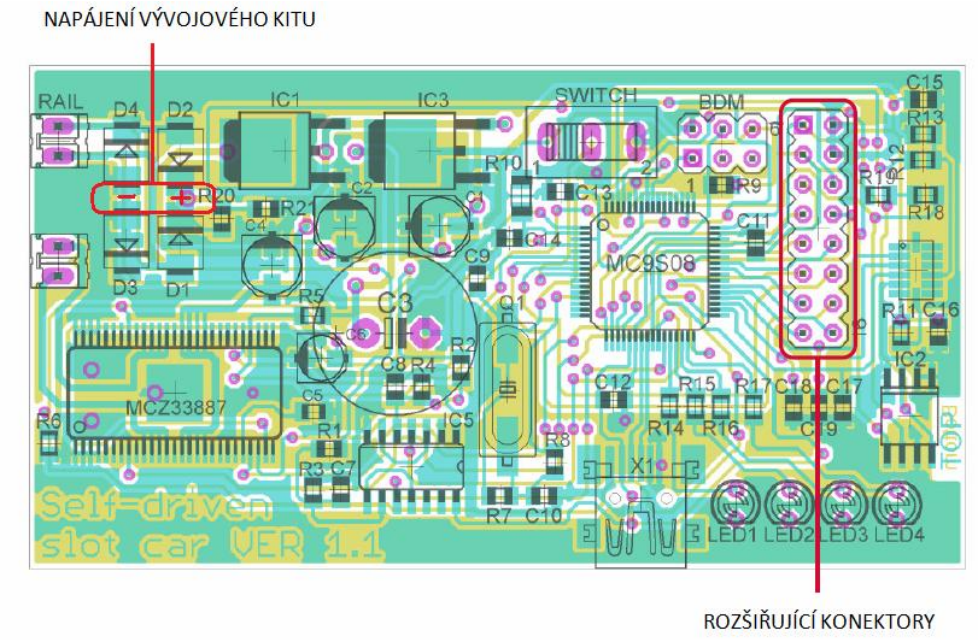
Obr. 20 Princip vestavěného řídicího systému

Kamera spolu s vývojovým kitem je určena pro detekci trajektorii dráhy. Algoritmus využívá pro detekci předem známý tvar dráhy, což usnadňuje rozpoznávání. Známým tvarem se rozumí tvar kolejnice. Dále algoritmus využívá poznatku množství odráženého světla zvlášť od kolejnice a černého povrchu dráhy.

Procesor vozu zajišťuje příjímání informace o detekci aktuálně zjištěné trajektorie dráhy. Následně se tato informace projeví na výkonu motoru. Pokud je vyhodnoceno, že před vozem se jedná o rovnou trajektorii dráhy, pak motor vozu využívá maximální výkon. Při vyhodnocení trajektorie zatáčky dochází ke zpomalení pohybu vozu a to na výkon motoru určený pro bezpečné projetí zatáčkou.

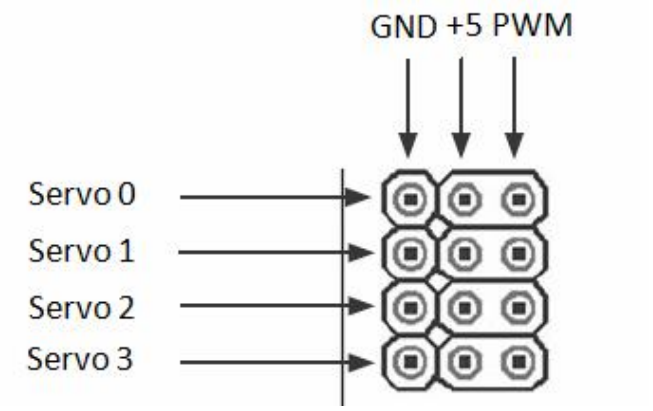
5.2. Propojeni kamery a modelu vozu

Správné propojení vývojového kitu kamery a modelu vozu je důležité pro komunikaci mezi jednotlivými částmi vestavěného řídicího systému. Důležitost této část systému se projevuje v rychlosti předávané informace a následné rychlosti a reakci elektroniky vozu, který ovládá motor zařízení. Pomalé předávání informace muže vést celý systém k neefektivnosti.



Obr. 21 Využití konektorů elektroniky desky vozu [6]

Propojení vývojového kitu s elektronikou vozu je znázorněno na Obr 21. Je zde zobrazeno přesné připojení kabelů nesoucí informaci o trajektorii dráhy. Využívá se propojení 2 pinů. Pin určený pro informaci rovné trajektorie dráhy je obsazen na pozici 10. Pin určený pro informaci trajektorie pravé, či levé zatáčky je obsazen na pozici 8. Také je potřeba propojení GND obou zařízení, to se nachází na pozici 2 a 4. Napájení vývojového kitu je realizováno za usměrňovačem na desce vozu. Kit je napájen z desky vozu 5DCV. Na Obr. 22 jsou zobrazena místa pro připojení datových kabelů z vývojového kitu kamery.



Obr. 22 Přípojná místa vývojového kitu [2]

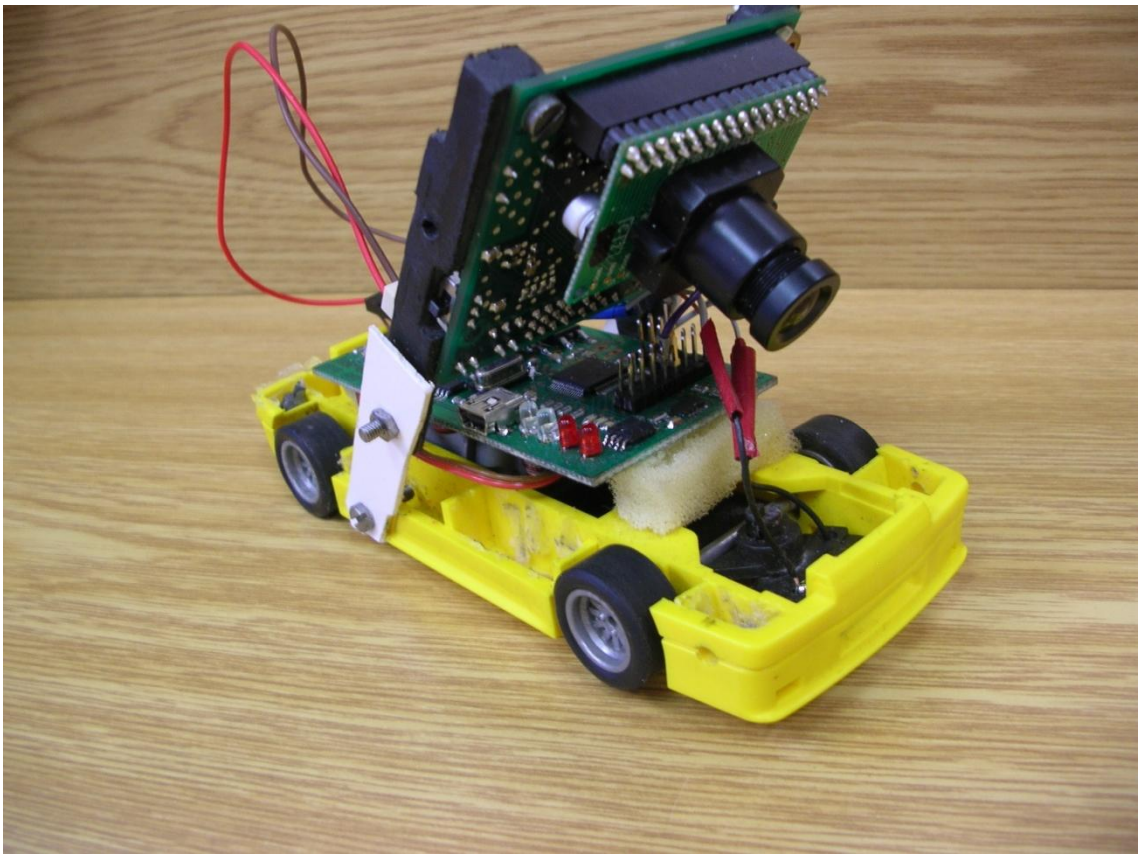
Část programu propojeni a přenos mezi částmi systému je umístěn v procesoru vývojového kitu. Jedná se o část určenou pro vysílání informace. Vysílaná data jsou vedena dvojicí datových kabelů. Informace o trajektorii dráhy je rozdělena na část pro informaci o rovné trajektorii a trajektorii zatáčky. Informace týkající se trajektorii zatáčky má dva stavy: levotočivá či pravotočivá trajektorie dráhy. Informace rovné trajektorie dráhy poskytuje stav: rovinka, či nikoli.

Příklad kódu části programu vysílající informaci z vývojového kitu kamery o detekci dráhy.

```
cc3_gpio_set_mode (0, CC3_GPIO_MODE_SERVO); //nastaveni modu serva
cc3_gpio_set_servo_position (0, 255);// predavana informace rovinka 1
```

Část algoritmu zajišťující zpracování přijímaných informací je podrobně popsán v následující kapitole 5.3.

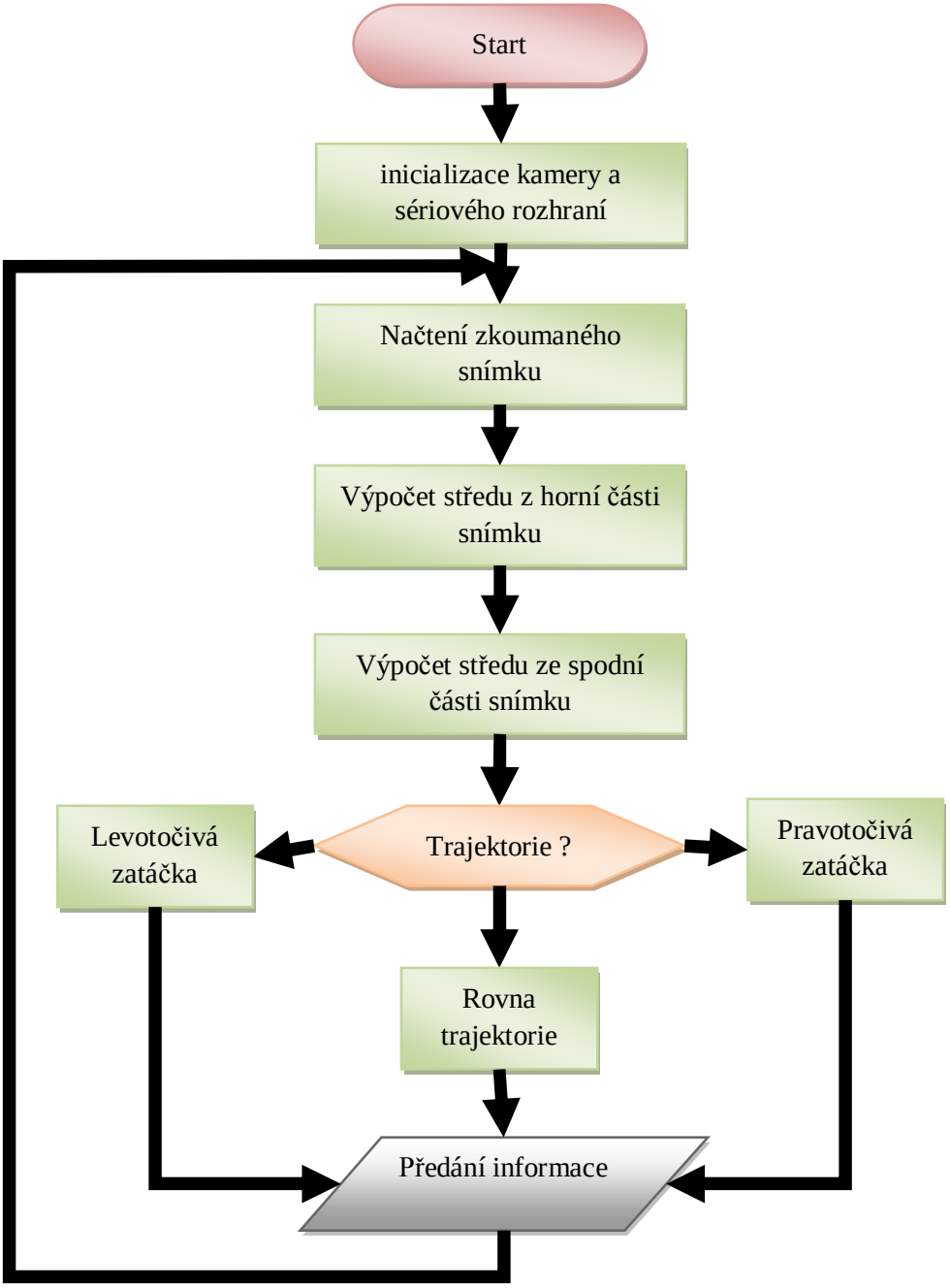
Celý vestavěný řídicí systém je zachycen na Obr. 23.



Obr. 23 Realizovaný vestavěný řídicí systém

5.3. Program pro vývojový kit

Následující algoritmus byl vyvinut pro část vestaveného řídicího systému a to pro vývojový kit kamery. Tento algoritmus zajišťuje detekci trajektorie dráhy. Program hledá střed kolejnic ve spodní a vrchní části snímku. Jde hlavně o pozici na ose x. Ta se následně porovnává.



Obr.24 Diagram algoritmu kamery

Algoritmus pracuje kromě standardních knihoven, i se speciálně určenými pro tento model kamery. Jsou to knihovny cc3.h, cc3_ilp.h. Zajišťují správný chod kamery. Na začátku programu jsou definované použité proměnné typu uint16_t. Například `uint16_t red`, využita pro matici obsahující hodnoty složky R. Následuje inicializace kamery, nastaveni sériového rozhraní pro komunikaci s PC. Také je zde nastaveno rozlišení kamery a barevné spektra.

```
cc3_filesystem_init ();// inicializace souboroveho systemu
cc3_uart_init(0, CC3_UART_RATE_115200,
CC3_UART_MODE_8N1,CC3_UART_BINMODE_TEXT); //konfigurace rozhrani
// stdout a stdin nejsou ve vyrovnavaci pamneti
val = setvbuf (stdout, NULL, _IONBF, 0);
val = setvbuf (stdin, NULL, _IONBF, 0);
cc3_camera_init ();// vlastní inicializace kamery
cc3_camera_set_colorspace (CC3_COLORSPACE_RGB);
cc3_camera_set_resolution (CC3_CAMERA_RESOLUTION_LOW
cc3_camera_set_auto_white_balance (true);
cc3_camera_set_auto_exposure (true);
```

Dále se provádí hlavní smyčka while, která zajišťuje neustálý běh celého zjišťovacího programu. V této části je například umístěno načtení snímku, zjištění velikosti jeho šířky a výšky. Nastavení všech barevných kanálů. Dále pak inicializace jednotlivých proměnných, z důvodu neustálého provádění smyčky. Například již zmíněné: `red = 0;`.

```
cc3_pixbuf_load ();     // nacteni snimku
img.width = cc3_g_pixbuf_frame.width;     // sirka snimku
img.height = cc3_g_pixbuf_frame.height;   // vyska snimku, osa y
cc3_pixbuf_frame_set_coi(CC3_CHANNEL_ALL); // nastaveni kanalu
```

Následuje jádro algoritmu pro detekci trajektorie dráhy. Procházení snímku je programově zmenšeno o 1/3 osy y od horní časti, což je v od hodnoty 47. Rovněž bylo programově zajištěno zmenšení osy x o 8,5 % a to z obou stran snímku. Důvodem bylo zkvalitnění detekce a odstranění nepotřebných částí snímku.

Tato část kódu reprezentuje zajištění počtu hodnot větších, jako předdefinovaná hodnota reprezentující hranici hodnoty jedna. Hodnota jedna určuje intenzitu výskytu jednotlivé barvy. Tato informace se dále v algoritmu zpracovává.

```
while (cc3_pixbuf_read_rows (img.pix, 1)) //prochazeni radku
{
      // kod pro vrchni cast sninku, prvnich 5 radku
      radek++;
      if (radek >= 47 && radek <= 51)//vyuziti prvnich 5-ti radku
      {
            bod1=0;     //promenna pro pocet 1
            for (uint16_t x = 15;x < img.width-15; x++)//velikost osy x
            {
                  cc3_get_pixel (&img, x, 0, &my_pix);
                  // vyber kanalu barvy a prirazeni do promenne
                  red = my_pix.channel[CC3_CHANNEL_RED];
                  green = my_pix.channel[CC3_CHANNEL_GREEN];
                  blue = my_pix.channel[CC3_CHANNEL_BLUE];
                  if (red > 157) // podmínka velisti (0,1)
                  {
                        bod1=bod1+1; // celkovy pocet 1
                  }
                  // … pro zbyvajici barvy stejne
            }
```

Část kódu určená pro hodnoty nacházející se na ose x jednotlivých řádků. Je opatřena podmínkou pracující s celkovým počtem zjištěných hodnot 1. Tyto hodnoty jsou základem pro určení středu. Následuje krátký výpis.

```
            bod1x = 0; //promenna pro hodnoty nachazejici se na ose x
            for (uint16_t x = 15;x < img.width-15; x++)//velikost osy x
            {
                  cc3_get_pixel (&img, x, 0, &my_pix);
                  red = my_pix.channel[CC3_CHANNEL_RED];
                  green = my_pix.channel[CC3_CHANNEL_GREEN];
                  if (red > 157)
                  {
                  bod1x=bod1x+1; // pocet 1
                  if (bod1x <= bod1/2 && radek == 47){vyslHorni1 = x;}
                  if (bod1x <= bod1/2 && radek == 48){vyslHorni2 = x;}
                  if (bod1x <= bod1/2 && radek == 49){vyslHorni3 = x;}
                  if (bod1x <= bod1/2 && radek == 50){vyslHorni4 = x;}
                  if (bod1x <= bod1/2 && radek == 51){vyslHorni5 = x;}
                  }
                  // … pro zbyvajici barvy stejne
            }
      }
      // kod pro spodni cast sninku, poslední 5 radku, jako predchozi
      // osa x stejna, osa y se pohybuje od 139 do konce snimku
}// konec while
```

Poslední částí je vyhodnocení o jakou trajektorii dráhy se jedná. Ze získaných dat se vypočte střed kolejnic vrchní části a spodní části snímku. Na základě těchto dvou hodnot program určí výslednou trajektorii. Podmínka obsahuje toleranci. Je potřebná z důvodu efektivního a bezchybného rozlišení roviny či zatáčky. Tato informace je dále posílaná na další součásti vestavěného systému.

Příklad výpočtu středu kolejnice horní části snímku.

```
vyslHorni=(vyslHorni1+vyslHorni2+vyslHorni3+vyslHorni4+vyslHorni5)/5;
```

Podmínka určení vychází ze zjištění absolutní hodnoty obou středů kolejnice. Každá výsledná možnost detekce dráhy obsahuje Posílaná informace je popsána v podkapitole 5.2.

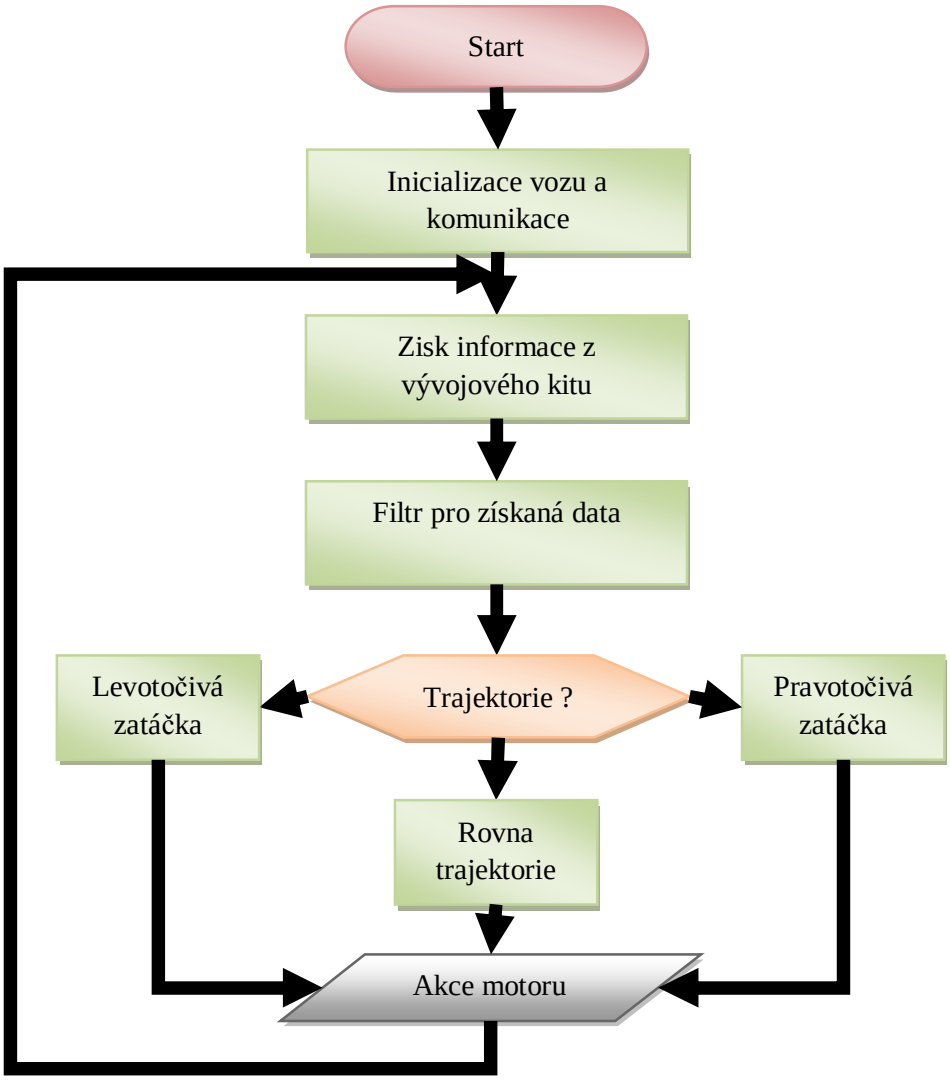
```
if (vyslHorni > vyslSpodni)
{
      rozdil = vyslHorni - vyslSpodni; // +
      if (rozdil<=25)   // mezni hodnota rozliseni rovinka / zatacka
      {
            // predavane informace o rovne trajektorii
      }
      else
      {
            // predavane informace o trajektorii pravotocive zatacky
      }
else // … podobne jako predchozi, jen obracena logika
{}
```

Po určení trajektorie dráhy se musí aktuální snímek uvolnit z paměti. Jinak by program pracoval se stále stejnými daty, nebo by také mohlo dojít k zhavarování algoritmu programu.

```
free (img.pix);
} //konec while (1)
```

5.4. Program pro model vozu

Tato část algoritmu je určena pro model vozu, která je součástí vestavěného řídicího systému. Algoritmus zajišťuje reakce motoru na přicházející informace z vývojového kitu kamery. Umožňuje také vizualizovat výsledek detekce pomocí led diod umístěných na desce vozu.



Obr. 25 Diagram algoritmu vozu

Je potřeba, aby byla zajištěna neustálá návaznost na aktuální data získávaná z vývojového kitu, proto je program umístěn ve smyčce for. Program obsahuje filtr, který zajišťuje vyhodnocení výsledku detekce z posílaných dat. Jelikož informace je posílána pomocí PWM. Následuje podmínka určená pro výkon motoru. Rovná trajektorie dráhy znamená maximální výkon motoru upravený pro konkrétní dráhu, naopak zatáčka přibližně poloviční výkon.

Na začátku programu vozu, se musí definovat použité knihovny, pro tuto část vestavěného systému. Tyto knihovny slouží například pro použití přerušení, nebo pro rozhraní zařízení.

```
#include <hidef.h>          /* povoleni preruseni */
#include "derivative.h"     /* deklarace rozhrani */
#include "slotcarutils.h"   /* utility */
```

Následuje definice konstant a proměnných. Tato část kódu obsahuje definice různých částí elektroniky desky. Například definice pro použití motoru, sběrnice, nebo led diod.

```
/* LEDs */
#define SET_LED_HL_ON          PTAD_PTAD0 = 1 /* bila krajni dioda ON */
#define SET_LED_HL_OFF         PTAD_PTAD0 = 0 /* bila krajni dioda OFF */
/* motor */
#define MOTOR_ENABLE           PTCD_PTCD4 = 1
#define MOTOR_DISABLE          PTCD_PTCD4 = 0
```

Dále pak následuje hlavní program včetně inicializace proměnných a celého programu. Inicializují se veškerá přídavná zařízení. Hodiny sběrnice se nastaví na 24 MHz a procesor na 48 MHz. Nesmí se opomenout inicializovat povolení rezistoru 0 až 2. Bez toho by mohlo dojít k poškození elektroniky vozu. Například:

```
PTDPE_PTDPE0 = 1;   // povolení pool rezistoru
```

Hlavní smyčka se neustále provádí. Je tím zajištěno neustálé zpracovávání informací přijímaných z kamery. Informace o trajektorii rovinky jsou přijímány pomocí definovaných vstupů: PTDD_PTDD0 pro informaci trajektorii roviny a trajektorie zatáčky PTDD_PTDD1.

Smyčka obsahuje programový filtr, určený pro zisk informace o trajektorii dráhy. Nejprve jsou hodnoty ukládány ze vstupu do proměnných. Poté jsou inicializovány příslušné proměnné určené pro výpočet.

```
a++;
if (a > 150)   // index pole
{ a = 0;
}
filtr[a]= PTDD_PTDD0;       // informace z rovne trajektorie
filtr2[a]= PTDD_PTDD1;      // informace ze zatacky
prumer = 0;
prumer2= 0;
```

Následuje výpočet hodnoty průměru vstupních hodnot. Jedná se o hodnoty posílané o rovné trajektorii dráhy a trajektorie zatáčky. Výpočet je prováděn ze 150 hodnot, následně se jednotlivé výsledky dále v algoritmu zpracovávají.

```
for (plus = 1;plus < 150;plus++)        // pocet nactenych hodnot
{
      prumer = prumer + filtr[plus];    // ulozeni vstupni hodnoty (0,1)
      prumer2 = prumer2 + filtr2[plus];
}
```

Tato část algoritmu vyhodnocuje, zda se jedná o rovinku, či zatáčku. Nevyhodnocuje se levotočivá, nebo pravotočivá trajektorie dráhy. To se realizuje v následující části kódu. Příslušné hodnoty se zaznamenávají do proměnných: rovinka, zatacka. Následuje ukázka podmínky určené pro informace přicházející o rovné trajektorie dráhy. Podmínka pro trajektorii zatáčky je podobná, liší se pouze použitou proměnnou: prumer2 a zatáčka.

```
if (prumer > 12)        // podminka pro rovinku
{
      rovinka = 1;
}
else
{
      rovinka = 0;
}
```

Předchozí části algoritmu se týkali zpracování příchozí informace o detekci dráhy z vývojového kitu kamery. Následující část algoritmu zajišťuje nastavení výkonu motoru vyhovující příslušné podmínce. Zároveň je realizována vizualizace jednotlivých částí dráhy. Pokud se jedná o trajektorii zatáčky, pak je nastaveno 45% výkonu motoru. Nastane-li levotočivá trajektorie dráhy, svítí prostřední diody, umístěné na elektronice vozu. Krajní diody jsou určeny pro pravotočivou trajektorii dráhy.

```
// vykon motoru
if (zatacka == 1) // prava zatacka
{
   motorVoltage = 2700;  // 45% vykonu motoru
   SET_LED_HL_ON;  // sviti krajni diody
   SET_LED_BR_ON;
   SET_LED_HR_OFF; // zhasnute prostredni diody
   SET_LED_BL_OFF;
}
else   // leva zatacka
{
   motorVoltage = 2700;  // 45% vykonu motoru
   SET_LED_HR_ON;  // sviti prostredni diody
   SET_LED_BL_ON;
   SET_LED_HL_OFF; // zhasnute krajni diody
   SET_LED_BR_OFF;
}
```

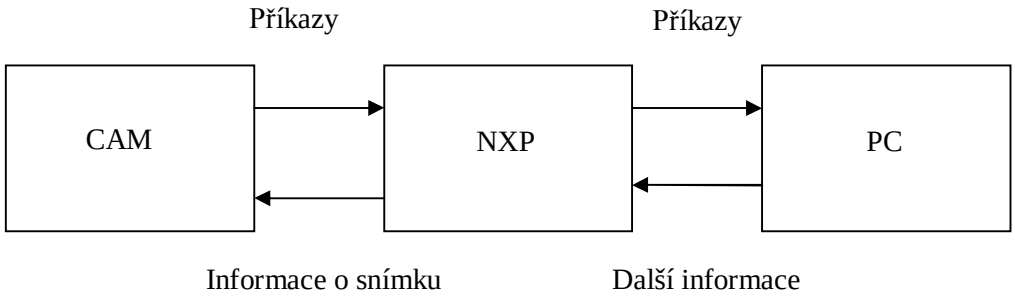
Pokud se jedná o rovnou trajektorii dráhy, svítí veškeré diody. Výkon motoru je nastaven na 60%, což je hodnota 3600. Maximální výkon motoru je definován v inicializaci programu.

```
if (rovinka == 1)
{
   motorVoltage = 3600;  // 60% vykonu motoru
   SET_LED_HL_ON;  // bila krajni – umisteni na desce
   SET_LED_HR_ON;  // bila prostredni
   SET_LED_BL_ON;  // revena prostredn
   SET_LED_BR_ON;  // revena krajni
}
```

6. Testování systému

6.1. Princip testování kamery

Testování kamery lze provádět pomocí propojení vývojového kitu a PC po sériovém rozhraní. Toto testování se uplatní při vývoji algoritmu rozpoznávání trajektorie dráhy. Příkazy, informace probíhající mezi kamerou a procesorem vývojového kitu, jsou zobrazovány na monitoru prostřednictvím příslušné aplikace. Je snaha o největší interakci s uživatelem. Tím se zvyšuje úspěšnost snadného odhalení případné chyby v programu.



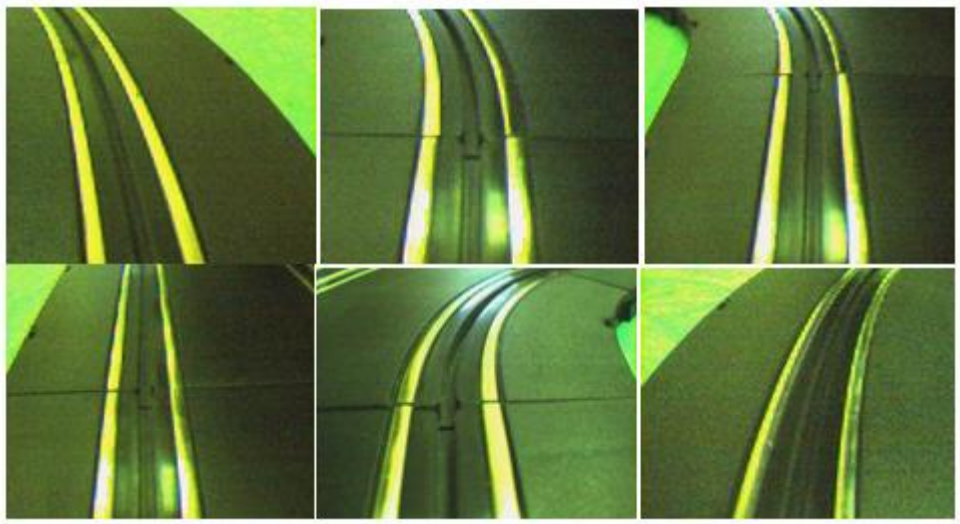
Obr. 26 Princip testování kamery

Aplikací, ve které je umožněno testovat aktuální algoritmus, je Hyper terminál. Tato aplikace je standardní součástí operačního systému Windows. Původně byl určen pro komunikaci pomocí modemu. Je nakonfigurován pro jakékoli zařízení schopné se připojit přes sériové rozhraní. Po importu programu do procesoru vývojového kitu a startu aplikace dochází k vizuální kontrole, jakou operaci daný algoritmus právě provádí. Na Obr. X je zobrazen výsledek. Ve spodní části je ukázka informace o závažné chybě vyvíjeného algoritmu. Tato chyba se projeví až při testování v procesoru, nikoli v kompilaci.

```
CMUcam3 v1.0b

Seriova komunikace funguje

Velikost snimku:
sirka snimku=352 vyska snimku=287

data abort!
0x000103D0
_
```

Obr. 27 Zobrazení testování

Testování různých situací na dráze, je důležité pro výběr nejspolehlivějšího rozpoznávacího algoritmu. Dráha není rovnoměrně nasvětlena, tudíž může docházet k nesprávnému rozpoznání tvaru trajektorie dráhy. Také byl nastaven vhodný úhel snímání obrazu pro kameru. Úhel kamery vůči dráze souvisí se získávanými daty. Pokud by byl úhel příliš velký, docházelo by k nesprávnému rozpoznání. Data z horní části snímku by obsahovala informaci o možné trajektorii zatáčky, přitom by se jednalo o rovinku. Tato omezení jsou řešena programově a to ořízntutím obrazu horní části o 1/3 a okrajů snímku podle umístění optiky kamery.



Obr. 28 Zobrazení testovaných vzorků snímku

Jednotlivé zobrazené vzorky snímku jsou pod zkratkami označeny v tabulce. L - levotočivá trajektorie, P - pravotočivá trajektorie, R – rovná trajektorie a Po – reprezentuje přechod mezi rovnou trajektorií a levotočivou trajektorii dráhy. Konkrétně se jedná o třetí snímek horního řádku. Algoritmy 1 a 2 rozlišují pouze Z – trajektorie zatáčky a R. Algoritmy 3 a 4 rozlišují veškeré možnosti.

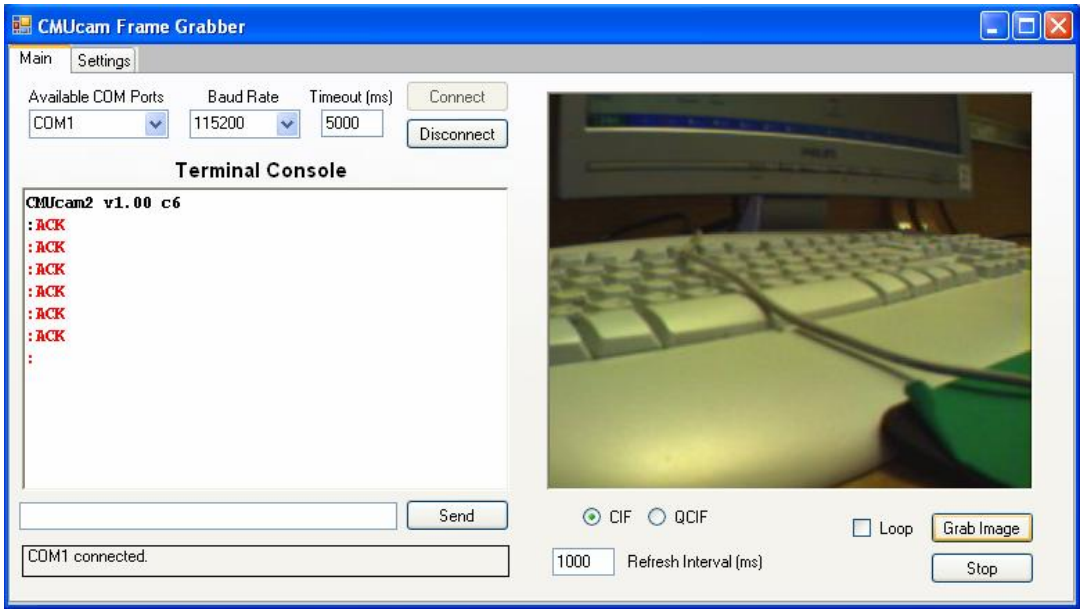
Skutečnost	1	2	3	4
L1	Z	Z	Z	L
L2	Z	Z	R	L
P1	Z	Z	Z	P
P2	Z	Z	Z	P
PO	Z	Z	R	R
R1	Z	Z	Z	R
R2	Z	Z	R	R

Tab. 2. Testování algoritmů detekce

Při testování různých systémů a způsobů algoritmů, byl vybrán ten nejspolehlivější. Mezi spolehlivé patřil čtvrtý algoritmus v pořadí, tento algoritmus byl dále vylepšován a jeho princip implementován do algoritmu vývojového kitu kamery. Veškeré testované algoritmy jsou popsány v kapitole 4. Kapitola 4 se zabývá simulací algoritmu pro analýzu obrazu.

6.2. Uživatelská aplikace pro zpracování obrazu

Uživatelskou aplikací, kterou je možno také testovat vyvíjený algoritmus a následně celý vestavěný řídicí systému je CMUcam Frame Grabber. Je určena pro tento typ kamer, ale i starší. Tato aplikace nabízí více možností, než již zmíněný Hyper terminál. Kromě vizuální kontroly probíhajícího procesu, umožňuje také různé nastavení snímaného obrazu v záložce Settings. Například přednastavenou volbu velikosti rozlišení. Plného rozlišení, nebo velikost snímku zmenšenou o ¼ původní velikosti snímku. Dále pak volbu barevného modelu RGB nebo YCbCr. YCbCr je způsob kódování RGB informací. Zobrazení barev závisí na aktuálním užití barev RGB modelu v signálu.



Obr. 29 Uživatelská aplikace [3]

Správné použití aplikace je podmíněno korektním nastavením. Prvním krokem je nastavení rozhraní "Available COM Ports"a nastavení rychlosti přenosu dat. Většinou na nejvyšší možné doporučené, což je hodnota 115200 Bd. Jedná se o rychlost, která udává počet změn stavu přenosu za jednu sekundu. Dále stlačením tlačítka "Connect, dojde k propojení aplikace a vývojového kitu kamery. Po této fázi následuje restart, nebo zapnutí napájení na vývojovém kitu uživatelem. Okamžitě poté začne komunikace. V levé části aplikace se zobrazí předdefinované výstupní informace. Uživateli je umožněno zadávat své vlastní příkazy a odesílat je pomocí tlačítka "Send". Tím je zajištěna interakce s uživatelem.

Tlačítkem "Grab Image" se zobrazí právě snímaný obrazu z kamery. Aplikace také automaticky ukládá snímaný obraz do PC. Tato funkce je užitečná při nastavování úhlu kamery, při snímání a ověření správnosti snímané plochy kamerou.

[3]

7. Zhodnocení výsledků a závěr

Cílem této bakalářské práce bylo navrhnout a realizovat vestavěný řídicí systém pro detekci trajektorie dráhy modelu auta na základě analýzy obrazu a na tuto trajektorii vhodně reagovat. Vestavěný systémy byl realizován na procesoru S08MJ32 firmy Freescale, který zajišťuje řízení vozu a na procesoru Philips NXP, který zpracovává data snímané kamerou.

Řešení vestavěného systému je rozděleno na dvě části. Nejprve byl vyvinut a implementován algoritmus určený pro procesor Philips ve vývojovém kitu kamery. Ten zajišťuje vyhodnocení dat s kamery a detekuje tak aktuální trajektorii auta. Z navržených algoritmů pro rozpoznání obrazu byl vybrán algoritmus, který vykazuje nejvyšší spolehlivost rozpoznání dráhy. Algoritmus je založen na porovnání vypočteného středu kolejnic v horní a spodní části snímku. Tento algoritmus byl naprogramován v jazyce C ve vývojovém prostředí CodeSourcery. Druhou částí je algoritmus pro vozidlo. Byl vyvíjen ve vývojovém prostředí Freescale CodeWarriorv a zajišťuje reakci motoru a LED diod na získávanou informaci předávanou z vývojového kiku kamery.

Omezení navrženého algoritmu pro rozpoznání vycházejí z mechanického umístění kamery na autíčku a světelných podmínek okolí. Jednalo se zejména o nasvětlení celé dráhy, po které se vestavěný řídicí systém pohyboval. Různé nasvětlení dráhy značně ovlivňuje vyhodnocovací algoritmus detekce dráhy. Velmi nepříznivě se projevují odrazy světla od dráhy a přímé osvětlení objektivu kamery světelným zdrojem. Tento problém byl částečně programově potlačen, a to ořezáním snímku dráhy. Vhodné umístění kamery a zvolený úhel sklonu umožnil snímat maximální plochu dráhy. To umožňuje snímat obraz dráhy ve vysoké kvalitě.

V rámci popsaných kritérii a omezení, pracuje vestavěný systém spolehlivě. Informace o detekované trajektorii dráhy jsou předávány z vývojového kitu kamery řídící desce modelu auta. Realizovaný vestavěný systém může být vylepšen doplněním algoritmu pro rozpoznání obrazu o vstupní filtr pro korekci jasové složky nebo stabilizací obrazu. Zvýšení spolehlivosti lze rovněž provést použitím referenčního algoritmu. Mechanicky může být systém zdokonalen použitím kamery s vyšším rozlišením nebo automatickým ostřením.

Vestavený řídicí systém může nalézt uplatnění v úlohách, kdy je potřeba detekovat trajektorii objektů. V praxi mohou být detekovány například okraje silnice nebo středová čára vozovky. Takové vestavěné systémy najdou uplatnění především v oblasti bezpečnosti, kdy mohou řidiče upozornit na vybočení z daného jízdního pruhu.

Literatura:

[1] *RGB* [online]. 2010 [cit 20-2-2010]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/RGB2010>.

[2] *Katalogový list kamery* [online]. 2006 [cit 2009-12-2]. Dostupné z WWW: <http://www.seattlerobotics.com/CMUcam3_datasheet.pdf>.

[3] *Průvodce instalací CMUcam3* [online]. 2006 [cit 2009-12-4]. Dostupné z WWW: <http://www.seattlerobotics.com/CMUcam3_sdk_guide.pdf>.

[4] *Cmucam podpora* [online]. 2004 [cit. 2009-12-28]. Dostupné z WWW: <http://www.cmucam.org/>.

[5] *FRC2009* [online]. 2004 [cit. 2010-1-28]. Dostupné z WWW: <http://hw.cz/FRC2009>.

[6] *Quick Start FRC2009* [online]. 2004 [cit. 2010-1-27]. Dostupné z WWW: <http://hw.cz/files/FRC2009-QuickStart.pdf>.

[7] *Faro podpora* [online]. 2004 [cit. 2010-1-27]. Dostupné z WWW: <http://www.autodraha-faro.cz>.

[8] *FRC2009* [online]. 2004 [cit. 2010-1-28]. Dostupné z WWW: <http://hw.cz/files/FRC2009.pdf>.

[9] *CodeWarrior* [online]. 2009 [cit. 2010-2-18]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/CodeWarrior>.

Přílohy:

Příloha I	Zdrojový kód simulace vytvořený v Matlabu
Příloha II	Zdrojový kód algoritmu vytvořený v CodeSourcery
Příloha III	Zdrojový kód algoritmu vytvořený v CodeWarrior
Příloha IV	Blokové schéma hlavních komponent kamery.
Příloha V	Seznam pinů kamery
Příloha VI	Referenční příručka cc3 (refman.pdf)
Příloha VII	Schéma plošného spoje desky vozu
Příloha VIII	Snímky celého systému
Příloha IX	Demonstrační video systému

Veškeré přílohy jsou v elektronické podobě přiloženy na CD.